

Integrated Simulation

Optimization Strategies and Logistical Process Control
for Production Planning based on
Collaboratively Maintained Queueing Models

D i s s e r t a t i o n
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

vorgelegt von

Dipl.-Inf. Martin Kramer
aus Mülheim a. d. Ruhr

und

Dipl.-Inf. Ingo Meents
aus Wilhelmshaven

genehmigt von der
Mathematisch-Naturwissenschaftlichen Fakultät
der Technischen Universität Clausthal

Tag der mündlichen Prüfung
14. Dezember 2001

Preface

Simulation is everywhere. Buying a new car one can be sure that its behavior in case of a crash has been simulated and optimized. Planes that we fly in have been thoroughly tested as computer models before the first flight, models have been tested in wind tunnels, pilots get trained in simulators.

In general, innovative products have become that complex that their behavior is quite often studied by simulation prior to their market entry. Complexity is also increasing in manufacturing processes. Before a decision about an investment in a new machine is taken the effects of different alternatives on capacity, work-in-process, or lead time are studied by means of simulation in order to choose the most promising one.

Is simulation only used in the development of new products, processes, or systems *before* something new is realized? In today's high tech manufacturing the border between ongoing production and the introduction of new processes gets blurred. Product life cycles of six to nine months (which are still decreasing) impressively show the speed at which changes are occurring. This is especially true for the production of read/write heads for hard disks. Every new product pushes the physical limits; new processes and machines have to be installed on the shop floor. For production management it is now critical to always have a valid model of the manufacturing line available to place the decision process on a solid basis. Thus, simulation more and more becomes a continuous process.

New questions arise quickly: How can the amount of data for such complex models be managed? How can other systems already containing data be incorporated? How is it possible to change parameters? Can results efficiently be distributed? How can the lead time of the simulation process be speeded up? Moreover, answers to those questions always have to be seen against the background of scarce resources.

Integrated simulation is a concept that provides answers to these questions by embedding simulation methods into the processes of production planning. It is important to apply a holistic approach, i.e. the whole process from data acquisition over the generation and management of simulation models to the handling of results has to be taken into consideration. *Integrated simulation* involves communication between a wide range of systems and users, if, for example, a model is to be built collaboratively by various persons, suitable front-end applications need to be available. Intranet technology and protocols have greatly facilitated the development and main-

tenance of such distributed applications. Communication between heterogeneous systems has effectively become possible by middleware technologies. The challenge is to consider simulation as an e-business process.

If such a system is available, new types of applications become possible. The construction of a consistent data warehouse of planning parameters enables the utilization of various operations research (OR) methods because one of the most serious problems of computational models is to get suitable parameters. Furthermore, fast analytical methods and simulation engines stretch the possible range of the application of the OR methods. For example, optimization based on large, up-to date models becomes feasible. Another set of new possibilities arises if performance measures of current simulation models are integrated with the statistical analysis of shop-floor-control systems. This allows, for example, the automatic generation of statistical process control charts for logistical processes.

Structure

This work presents the generic model of *integrated simulation* and describes how this concept can be derived from new demands of today's production management problems. The design of EPOS, a system which features many requirements of *integrated simulation*, is presented. These two parts have been coauthored by Martin Kramer and Ingo Meents. In part 3 *Advanced Planning and Optimization* two further topics are covered in-depth. These have been exclusively developed and written by either of the authors:

Ingo Meents: Chapter 10 Optimization

Martin Kramer: Chapter 11 Integration with Shop-Floor-Control

Acknowledgements

It's been quite a journey to get to this point, and we owe many thanks to several people for a great deal of different things. First of all, the authors would like to thank our dissertation chair, Prof. Dr. Thomas Hanschke, the initiator of IBM's doctorate scholarship program. He has been the greatest supplier of inspiration and encouragement for us and the whole team. Thank you for all the great brainstorming sessions and new ideas.

We would also like to thank the members of the committee, Prof. Ph. D. Gerhard R. Joubert, Prof. Dr. Ingbert Kupka, and Prof. Dr. Jürgen Fertig for their time and effort.

The IBM Deutschland Speichersysteme GmbH has given us the possibility of taking part in the doctorate program and has put the required means at our disposal. We would like to thank Dr. Walter Meizer and Dieter Münk representing the company. For lasting and ongoing support of the EPOS project especially Klaus-Jürgen Rünger, Helmut Rink, Dr. Friedrich Gerken, Thomas Bück, Ralf Blasek, and Rolf-Dieter Bürk deserve our thanks.

Simulating the Mainz wafer line would not have been possible without the help and support by our colleagues Bruno Hermann, Andreas Fröse, Hans-Jürgen Göppert, Jürgen Tschirner, Hans Dötsch and the whole staff of the line control & logistics department. We are grateful to Bernd Cohrs who does a great job administering EPOS, thank you for your tireless work and dedication. Peter Domokos and Krisztian Bolygo worked with us to analyze and simulate the HDD assembly line in Szekesfehervar, Hungary.

The EPOS team deserves our gratitude. It has been a pleasure for us to work in such a dedicated team. Dr. Horst Zisgen has laid the foundation with his work. Thanks to Michael Frank for managing the team in Clausthal. Thomas Klein, Jens Rehaag, Lars Dohse, Sören Paul, and Horst BIRTHELMER have worked indefatigably to make all the ideas work.

We have been lucky to have a great technical support team at our side. Representing their team we would like to thank Sven Oehme, Matthias Dietz, and Stefan Schorn for their fast and non-bureaucratic help.

Personal Acknowledgements by Ingo Meents

Very special thanks go out to my family and my girl-friend Andrea whose support, encouragement, patience, and confidence in me throughout the years made this dissertation possible.

Personal Acknowledgements by Martin Kramer

There is one special person that deserves my deepest thanks and respect. Thank you, Myriam, for all your love and support, for our conversations and your encouragement, and last but not least, for being my friend. I also have been very fortunate to be supported by my family throughout all the years.

Ingo Meents und Martin Kramer

Mainz, November 2001

Contents

I. Integrated Simulation	1
1. Introduction	3
1.1. The Challenge	3
1.2. Solution Methodology	4
1.3. Production Management	7
1.3.1. Hierarchical Planning	7
1.3.2. Production Systems	9
1.3.3. Capacity Planning	11
1.4. Modeling and Simulation	17
2. The Model of Integrated Simulation	23
2.1. Analysis of the Current Situation	24
2.2. Features Required for Integrated Simulation	26
2.3. Literature Review	33
2.4. Collaborative Creation of Simulation Models	36
2.4.1. User Roles	36
2.4.2. Distributed Parameter Input	41
2.5. Automatic Model Generation and Transformation	46
2.6. Generation of the Real-World Model	49
2.7. Transformation of the Real-World Model	51
2.7.1. Modeling Techniques	53
2.7.2. Parameterized Model Transformation	58

2.7.3. Processing Incorrect and Incomplete Data	60
2.7.4. The Simulator	62
2.8. Model Validation	63
2.9. Business Process Capacity, Lead Time, and Work-in-Process Planning	66
2.10. Summary	69
3. Queuing Network Analysis	73
3.1. Introduction	74
3.2. The Model	76
3.3. Input Parameters	77
3.4. Demand and Bill-of-Materials	79
3.5. Number of Visits	80
3.6. Product Mix	81
3.7. Probability of Good Parts	81
3.8. Number of Visits at Work Centers	82
3.9. Service Times at Work Centers	83
3.10. Completion Times at Work Centers	83
3.11. Batch Processing	85
3.12. Network Decomposition Method	87
3.13. Performance Measures	89
3.14. Work-in-Process and Lead Time	90
3.15. Overall Performance Measures	91
3.16. Process Step Performance Measures	92
3.17. Remaining Lead Time	93
II. EPOS - A System for Integrated Simulation	95
4. System Architecture	97
4.1. Introduction	97
4.2. System Overview	99
4.3. Core Components	102
4.3.1. Company Structure	102
4.3.2. Work Centers	106
4.3.3. Products	114
4.3.4. Volume Plans	118
4.3.5. Routing	118

4.3.6. Staffing	123
4.4. The Central Plan Parameter Database	125
4.4.1. Integration of Spreadsheets	126
4.4.2. Interfaces to Shop-Floor-Control Systems	127
5. Tool-Parameter-Sheets	129
5.1. Overview	129
5.2. Deployment	130
5.3. Data Input — The Java Applet	132
5.3.1. Manufacturing Input	133
5.3.2. Maintenance Input	134
5.3.3. Engineering Input	135
5.3.4. Staffing Input	137
5.3.5. Security Aspects	137
5.4. Tool-Parameter-Sheets — The Notes Database	138
5.5. Use of Different Database Paradigms	140
5.5.1. The Benefits	140
5.5.2. Replication Between Databases	142
5.6. Integrity Checks	143
5.7. Business Process for Plan Parameter Input	145
6. The Simulator	147
6.1. Simulation Server	147
6.1.1. Deployment	148
6.1.2. Static Structure	151
6.1.3. Performance Measures	155
6.2. EPOS Analyzer	156
7. Automatic Model Generation	161
7.1. The Simulation Environment	161
7.1.1. The Model Generator	162
7.1.2. Simulation Requests	163
7.1.3. Simulation Runs	167
7.2. Automatic Model Generation	171
7.2.1. Phases of the Automatic Model Generation and Sim- ulation	171
7.2.2. Loading the Real-World Model	172
7.2.3. Generating the Simulation Model	177

7.2.4.	General Schema of the Model Generation	177
7.2.5.	Creating the Routing	182
7.2.6.	Modeling Transportation	188
7.2.7.	Special Work Center Types	191
7.2.8.	Examples	195
7.2.9.	Different Planning Scenarios	204
7.2.10.	Multi-Process Production Lines	204
7.3.	Automatic Simulation	208
7.3.1.	Simulating the Model	208
7.3.2.	Saving the Results	209
7.3.3.	Calculating Line Profiles	213
7.4.	Processing Inconsistent and Incomplete Data	214
7.4.1.	Principles	215
7.4.2.	Incomplete Data	215
7.4.3.	Inconsistent Data	217
7.4.4.	Event Logging	220
7.5.	Summary	222
8.	Reporting	223
8.1.	Deployment	224
8.2.	Types of Reports	225
8.3.	Input for the Simulation	228
8.3.1.	Volume Plans	228
8.3.2.	Simulation Model	228
8.3.3.	Consistency Checks Prior to Simulation	230
8.4.	Monitoring the Process of Model Generation	232
8.5.	Simulation Results	233
8.5.1.	Detailed Standard Simulation Report	234
8.5.2.	Commented Standard Report	241
8.5.3.	Accessing Simulation Results	244
9.	Administration	247
9.1.	Requirements	247
9.2.	Levels of Administration	250
9.3.	The EPOS Administrator	254
9.3.1.	Overview	255
9.3.2.	Applications	255
9.4.	Technical Administration	259

III. Advanced Planning and Optimization	261
10. Optimization	263
10.1. Problems and Methods	263
10.1.1. Introduction	263
10.1.2. Linear Programming	268
10.1.3. Quadratic Programming	269
10.1.4. Evolutionary Algorithms	270
10.1.5. The Goal	272
10.2. Planning Manufacturing Output	273
10.2.1. The Planning Task	273
10.2.2. Basic Model for the Optimum Product Mix	274
10.2.3. Extension of the Model	277
10.2.4. Integrating the Product Mix Optimization	279
10.2.5. Example	286
10.3. Routing/Load Optimization	287
10.3.1. Junctions	288
10.3.2. A Quadratic Program for Routing Optimization	293
10.3.3. Objective Function	295
10.3.4. Examples	299
10.4. Optimization of Performance Measures by Evolutionary Search	308
10.4.1. Simulation and Optimization	308
10.4.2. Chromosome Coding	311
10.4.3. Initialization Schemes	313
10.4.4. Genetic Operators	315
10.4.5. Constraint Handling Techniques	319
10.4.6. Objective Functions	322
10.4.7. Experiments	328
10.4.8. Results	344
10.5. Conclusion	344
10.5.1. Results	345
10.5.2. Outlook	347
11. Integration with Shop-Floor-Control	351
11.1. Introduction	351
11.1.1. Overview	352
11.1.2. The Goal	354
11.1.3. Review of the Literature	354

11.1.4. Systems	362
11.1.5. Analysis of Stochastic Processes	365
11.2. Logistical Process Control (LPC)	384
11.2.1. Overview	384
11.2.2. Control Charts for Process Time at Process Steps . .	387
11.2.3. Methods for Estimating the Variance of the Sample Mean	399
11.2.4. Experimental Results	407
11.2.5. Control Charts for Waiting Time	413
11.2.6. Other Logistical Processes	417
11.2.7. Design of a Decision Support System	419
11.2.8. Results	422
11.3. Work-in-Process Forecast	426
11.3.1. Calculating the Forecast	426
11.3.2. Matching Outcome and Demand	429
11.3.3. Presentation	429
11.3.4. Results	432
11.4. Model Validation	433
11.4.1. Locally Optimized Batch Size	433
11.4.2. Batch Size Correction to the Transport Batch Size . .	437
11.4.3. Results	439
11.5. Conclusion	439
11.5.1. Summary	439
11.5.2. Review of the Current Use of Shop-Floor-Control Sys- tems	441
11.5.3. Requirements for Future Shop-Floor-Control Systems	443
11.5.4. Outlook	444
12. Conclusion	445
12.1. Summary	445
12.2. Outlook	447
IV. Appendices	451
A. Database Tables of Sample Model	453
B. Abbreviations	459

C. Interface of the Simulation Server	463
D. Legend for Work Center Parameters	473
E. The IBM Corporation	475
E.1. IBM Deutschland GmbH	476
E.2. The Manufacturing Plant in Mainz	477
F. Storage Systems Technology	479
F.1. Thinfilm Discs	480
F.2. The Wafer Process	481
F.3. The Slider Process	481
Bibliography	483
List of Figures	507
List of Tables	513
 Curriculum Vitae (Martin Kramer)	 515
 Curriculum Vitae (Ingo Meents)	 519

Part I

Integrated Simulation

Chapter 1

Introduction

1.1. The Challenge

It is well accepted that production planning contributes essentially to a company's success. Shorter product life cycles and growing competition demand a planning method that allows to react instantly and efficiently to changes in the production environment. Especially if the environment is highly stochastic because of machine breakdowns, quality problems, yield loss, set-up requirements, demand and process flow changes, operator unavailability, etc., simulation is an appropriate means to aid the planning process. But the more the complexity of the production environment increases, the more difficult it gets to generate and maintain appropriate simulation models. Traditionally, simulation experts using simulation systems on stand-alone work stations have to collect the required parameters by paper based procedures like machine parameter sheets.

Production planning for complex high-volume manufacturing lines is a rather difficult task. Especially in the area of semiconductor manufacturing, production systems consist of several hundred machines grouped into distinct work centers. The process sequence is specified by a consecutive list of process steps, and products have to undergo several hundred of operations before completion. The largest of the production lines examined during the work on this thesis contains up to 150 work centers, 20 products of

approximately 400 operations each. All these entities have several attributes that have to be taken care of for a reliable planning.

The most striking characteristic of such production systems is the highly re-entrant flow, i. e. parts make multiple visits to work centers as successive layers are added. Unrolling these loops on the shop floor in order to obtain straight flows would be far too capital-intensive. Moreover, the process flow is disturbed by production parts sent to rework. Due to the leading-edge technology scrap rates are often quite high. Thus the number of wafers in transport batches is varying in the course of production. All these characteristics turn proper analysis and planning into a very difficult task.

The fast pace of technological advance, especially in semiconductor and hard disk fabrication, leads to short product life cycles. Today industry faces product life cycles of about six to nine months which are still decreasing. New products often require new manufacturing processes and machines on which these can be performed. Together with fast changing demands production planning on the tactical level like capacity planning is becoming an ongoing process.

The goal of this thesis is to design a concept as general as possible to support various tasks of production planning, not only during the design phase of a manufacturing system but also during its operation. The appropriate planning methods and information system technology that allow an efficient use and implementation of such a system have to be identified. Moreover, existing business processes have to be considered and re-engineered so that newly designed processes are not in competition to the existing ones but either improve and replace them or cooperate with them.

1.2. Solution Methodology

The solution to the challenging tasks presented in this thesis can be divided into two parts. First of all, an information system is designed that allows to acquire all planning parameters needed to construct a model of the complex production lines to be analyzed. The second step is to incorporate advanced planning methods that allow for a fast and accurate production planning.

The concept of the information system is based on the characteristics of semiconductor/storage manufacturing: medium to high volumes and large, complex production lines. However, these characteristics do not only apply to the production of read/write heads for hard disk drives, but also to pro-

duction systems in other industries like steelworks. The time frame chosen is mid to long term planning. This setting motivates the use of queueing network analysis as due to the high volumes the statistical assumptions are justified and the analysis is fast.

The problem lies in the construction and maintenance of such queueing network models. Modern information technologies help to tackle these problems: a persistent object model of the production lines is stored in a database. People in charge of machines are assigned the responsibilities to administer parameters. The queueing networks are automatically generated from this database such that these models can be updated or recreated any time the environment changes. This allows an ongoing use of simulation. The results of the analysis are distributed within the company to the appropriate planners. The thesis in detail describes the modeling approach and the processes needed to build queueing network models with the help of a whole company, either people or systems.

After a system which combines information technology and sophisticated mathematical methods has been established, two further topics are examined. The first one covers optimization techniques. Queueing networks only allow to analyze scenarios defined a priori. They can help to estimate future utilization, work-in-process, and lead times. But they do not try to find improved or even optimal scenarios. Moreover, the construction of queueing models from factory data leaves certain degrees of freedom to the generator. The questions covered in this context are the determination of optimal product mixes in deterministic and random environments, the definition of optimal distributions for routings probabilities at junctions in the process flow, and several general goal programming tasks, e.g. which minimum investments allow to achieve a certain lead time etc. The techniques applied are mathematical programming and evolutionary algorithms.

The second topic is the integration of queueing network analysis and shop-floor-control systems. Modeling and analyzing a system is only the first step. The next step must be to compare the results to the actual performance of the production system. For large production lines this requires a systematic approach as thousands of parameters have to be compared. The comparison offers two opportunities: On the one hand the validation of the model and on the other hand problem detection by means of logistical process control (LPC). This is the application of statistical process control for logistical processes, the results are control charts comparing the target value from the simulation studies with the observed performance of the produc-

tion system. The basis of these charts are statistical tests. The integration of shop-floor-control and simulation is not limited to comparisons, though: Model parameters can be estimated from the observed processes and by using current data from the shop floor simulation can be employed to generate operational plans.

The thesis is structured as follows: Chapter 2 analyzes the current situation and introduces the term *integrated simulation*. Some drawbacks and problems are used to motivate the requirements that finally lead to the thorough concept.

The mathematical backgrounds and a special model for queueing network analysis are presented in chapter 3. Based on the concept a prototypical system called EPOS that realizes most of the defined requirements is developed in part II. A description of the system's core components can be found in section 4.3.1. As special emphasis is put on the manual data input chapter 5 explains how the input parameters are structured such that the responsibility of maintaining that data can be distributed. The corresponding processes are explained and the technology that allows an easy roll-out of the software is presented. The simulation environment, i.e. the simulation server and the interactive user client EPOS Analyzer, is shown in chapter 6. This is followed by a description of the automatic model generation and transformation in chapter 7. The distribution of the results and the types of reports that are the outcome of the simulation system are shown in chapter 8. Chapter 9 describes the system from the viewpoint of system administration, the different administrators needed and the tools available for them.

The next part of the thesis focuses on advanced planning strategies. Chapter 10 introduces optimization methods to plan optimal product mixes, to balance the load on certain work centers in the simulation model and to find improved settings of the production environment with respect to lead time, work-in-process, and financial decisions. Chapter 11 shows how the results of the queueing network analysis can be combined with actual data from shop-floor-control systems. This includes validation methods, logistical quality charts, and a forecast algorithm to estimate the output dates of single wafers from the production line.

The appendices contain technical information like database tables filled with data of a sample model, a list of abbreviation used in the thesis, and the complete definition of the interface of the simulation server.

In the following this introductory chapter explains the basics of production management like hierarchical planning and different types of manufac-

turing systems, and simulation methods in order to allow to position *integrated simulation* within the framework of traditional planning approaches. Emphasis is put on the difference between discrete event simulation and queueing theory. Moreover, the company that enabled this work, the IBM Deutschland Speichersysteme GmbH, and its manufacturing processes that motivated many of the ideas within this thesis are presented.

1.3. Production Management

Production management covers all aspects that are needed for an efficient and competitive organization of a production process. This includes topics like production planning and control, scheduling, financial aspects, demand planning, storage planning, etc. Subject of all investigations is the production system that contains resources like workers and machines contributing to the output of finished goods. Typologies of production systems are introduced in section 1.3.2.

Production planning is usually performed on different aggregational levels concerning the time span of decisions. The common approaches are described in next section.

1.3.1. Hierarchical Planning

Planning decisions for the design and the operation of a production system are usually hierarchical in nature. Most authors distinguish between *strategic*, *tactical*, and *operational* level [Ada97, Ste90, DPL94, Swa00]. The different levels and the corresponding decisions are summarized in figure 1.1.

At the highest level are strategic decisions that deal with the general policy of a company and thus usually involve top-level management. These decisions are broad in scope and involve choice of products and services, new facilities and the size and locations of new plants, etc. The information is highly aggregated, and its source is to a large extend external. For the most part, strategic decisions have long-term implications for the whole organization and set constraints for lower decision levels.

Decisions at the tactical planning level are narrower in scope than strategic decisions. Planners try to assure that resources are obtained and used efficiently in the accomplishment of the organization's objective. These decisions usually involve the consideration of a medium range time horizon. For example, normally production plans are determined over a 3 to 18 months

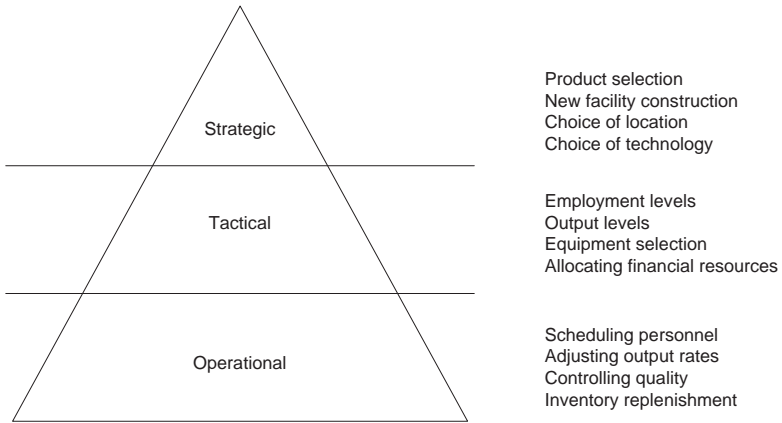


Figure 1.1.: Hierarchy of operations management decisions [Ste90]

time horizon. Sometimes an aggregation of products into families is useful. In this production plan, the quantities to produce at each period of the horizon (a week or a month) are calculated so that the demand is satisfied while the production and holding costs are minimized. The decisions to be made in this scenario are how much is to be produced in each period.

Other related typical decisions to be made are the utilization of equipment and of regular and overtime workforce, allocation of capacity resources to products (product families), planning of output rates and inventory levels, etc. Tactical decisions are made within the framework established by strategic decisions and their source of information is internal and external.

At the lowest level of the hierarchy operational decisions are taken which can be characterized by a shorter time horizon — usually several days up to a couple of weeks. The information used at this level is very detailed and its source is largely internal. If the goal of the tactical level is given on an aggregate level, a disaggregation with respect to the current state of the production system is necessary first.

Operational decisions involve such activities as scheduling of equipment and personnel to achieve the goals specified on the tactical level, adjusting production rates, handling equipment breakdowns, absenteeism, shortages, inventory replenishment, quality control, expediting and processing of or-

ders, vehicular scheduling, forecasts of work-in-process and lead times, etc. Operating decisions are made within the framework established by tactical decisions [Ste90].

Usually, this hierarchical model is not always exact, since in some cases decisions can be taken more or less early in the hierarchy. Most of the time, decisions at each level are taken in sequence, i.e. in a top-down approach, while it is not guaranteed that the production plan computed at the tactical level is an achievable objective for the operational level. This partly explains large inventories and delivery delays for customers in today's factories. Indeed, no matter how sophisticated the scheduling method used at the operational level is, it will be of a limited efficiency if the objective given by the tactical level is not realistic.

1.3.2. Production Systems

Many different typologies of production systems exist. The one given in [DPL94] distinguishes between four different typologies according to the number of parts to produce:

- *Unitary Production.* The large size of the finished products imposes a production of small quantities, and the main task is to provide all the production resources when and where necessary. In this type of project management techniques play an important role. Examples of this kind of production are rockets, ships, aircrafts, bridges etc.
- *Production in small and medium series.* The finished product has a smaller size, and the manufacturing process usually takes place in a workshop. The problem is not to minimize the total time for producing one item but rather for the whole production and a related goal is to minimize the waiting time.
- *Production in large series.* In the case of large numbers of similar products to be produced in the same period of time, or when the number of different products is limited, it may be interesting to organize transfer lines of production where the resources are distributed according to a fixed ordering. In doing so, one can reduce the waiting lines in front of the resources. The main problem is thus to create equilibrate transfer lines (line balancing).

- *Continuous production.* In this case, the transfer line is continuous, i. e. any waiting time between the resources is excluded. This type of production occurs for example when one has to manipulate liquids or gas. Again, a good line balancing is necessary for efficiency.

Another typology for production systems can be the type of organization. With respect to this typology, one can distinguish between four different types:

- *Stationary manufacturing.* For stationary manufacturing the equipment and materials are taken to the construction site. Typically, this is the case for large objects that are built only once for a special customer. Examples are ships, bridges, buildings, etc.
- *Shop floor production.* In a job shop system the manufacturing environment is organized according to the function/process of work centers. Parts are routed on the shop floor according to the processing requirements.
- *Flow shop production.* In a flow shop manufacturing system the work centers are ordered according to the process flow of the products. The operations are usually quite short and approximately of the same length. This type of organization is typical of the high-volume production of standard products.
- *Cellular production.* For this organization type the set of all products is divided into subsets, so-called part families. Each family is processed on a subset of the work centers, a so-called machine cell. A machine cell and its corresponding part family form a manufacturing cell. Another name is *flexible manufacturing system*.

The connection between both typologies is shown by figure 1.2. The x-axis represents the number of parts per time unit whereas the y-axis shows the degree of the products' specialization. For example, the unit production of a ship is highly specialized (with respect to the goods produced) and very small in volume, i. e. normally just one single ship for a special customer. On the other hand, the production of TV sets is a mass production with rather low specialization, i. e. a huge number of identical TV sets is sold to thousands of customers.

These categories will be used later in figure 2.11 to explain the possible uses of the concept presented in the next chapter.

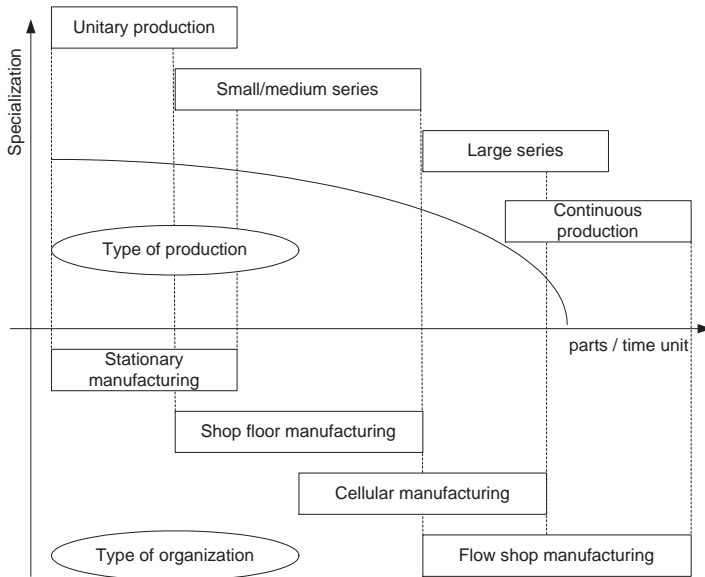


Figure 1.2.: Typology of production systems

1.3.3. Capacity Planning

Capacity planning is a very important task at each of the three levels (operational, tactical, strategic) of production planning. This section introduces the most important terms of capacity planning. Table 1.1 summarizes capacity-related topics and associates them to each of the planning levels. Capacity planning is the function of determining the level of capacity needed to achieve scheduled production, comparing this with available capacity, and planning necessary adjustments in capacity levels or schedules. Critical elements of capacity to be planned include labor, machine hours, facilities, warehouse space, and engineering.

Capacity planning is of crucial importance in achieving successful results in production planning and control. If insufficient capacity to achieve the production schedule is provided, the results will be shortages, failure to meet production targets, overdue shipments to customers, frustrated production

	Resource planning	Rough-cut capacity planning	Capacity requirements planning
Planning horizon	long up to 5 years	long-medium 1 to 3 years	medium-short 1 year
Degree of detail	company	production line	work center
Time buckets	months or quarters	weeks	weeks
Planning purpose	plant expansions, special equipment, major change of workforce	product mix, standard equipment, sub-contraction	overtime, routings, flow control, re-allocation of work force
Inventories	no	no	yes (netted plan)

Table 1.1.: Hierarchy of capacity planning [Smi89]

managers, and a loss in confidence in the formal (calculation) system. As it becomes obvious that stated production schedules cannot be accomplished with the resources provided, plans produced by the formal system are abandoned in favor of ad hoc decisions on the factory floor. On the other hand, if more resources are provided than are needed, the result is low utilization of resources, manufacturing inefficiency, high costs, and reduced profit margins [Smi89].

The following terms are used in capacity planning:

- *Capacity* is the rate at which a productive system (worker, machine, work center, department, plant) can produce. This is defined in terms of units of output per unit of time. Typical units for the capacity are daily going rate (DGR), weekly going rate (WGR), or the capacity with respect to different shift models.
- *Queue* (or backlog) is the amount of work (in parts) waiting to be performed in a productive system.
- *Load* (arrival rate, start rate) is the amount of work scheduled to be performed by a productive system in a specified time period. The load

is given in parts per time unit.

- *Required capacity* is the capacity needed to meet the demanded volumes
- *Maximum capacity* is the theoretical or potential capacity of the production system. This is based on the assumption of ideal conditions such as three-shift, seven-day-per-week operation with no downtime.
- *Demonstrated capacity* is the rate of output that can be expected on experience, taking into account current and planned levels of resources such as manpower, overtime, and the number of shifts.
- *Cycle time restricted capacity*. The cycle time restricted capacity links cycle time estimates to capacity planning (see [RFN00]). Usually, the cycle time increases non-linearly with increasing system load to infinity, i.e. while the load approaches 100%, the corresponding cycle time increases dramatically. Of course, this is not desirable. Setting an upper limit on cycle time imposes an upper limit of the system's capacity. The relationship between load and cycle time shows the so-called line profile (cycle time vs. load curve) in figure 1.3. Once the profile has been plotted, the desired cycle time determines the maximum system load (the inverse of the profile). The cycle time can be expressed in terms of the raw process cycle time by introducing the factor X , i.e. cycle time = $X \cdot$ raw process cycle time.

For the planning of the capacity of a production system utilization plays a very important role. In general, the *utilization* of a tool¹ is the ratio of hours worked to hours available. In today's semiconductor manufacturing systems, machines are normally operated 24 hours a day, seven days a week as machines are usually very capital-intensive. Taking also certain capacity loss factors into account gives rise to the following definitions of utilization:

- *Net utilization*. Net utilization is the ratio of hours worked to total time scheduled for production. This measure does not take any loss factors for the capacity into account.
- *Total utilization*. For the calculation of the total utilization the total time scheduled is reduced by a certain amount of time due to capacity losses. These capacity loss factors include in decreasing order of importance (based on [RFN00]):

¹In semiconductor manufacturing machines or servers are often referred to as *tools*.

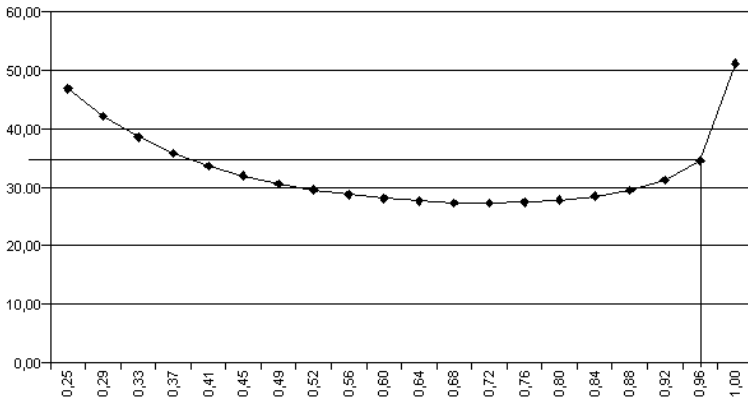


Figure 1.3.: Cycle time line profile

- ▷ Down time. The most striking effect on a production system's capacity is caused by down times of the machines. Two types of down time can be distinguished: scheduled preventive maintenance and random breakdowns. Both reduce the available time for processing by a certain percentage. In addition, unscheduled random failures introduce more variability in the production system, leading to longer lead times.
- ▷ Yield loss. In general two situations have to be distinguished depending on where the major yield loss actually occurs: If the major yield loss occurs behind the bottleneck, the time of the parts scrapped behind the bottleneck is completely lost and cannot be recovered. Yield loss in front of the bottleneck increases the variability in lot arrivals, and thus increases cycle times and decreases the cycle time constrained capacity.
- ▷ Set-up. The time to set up a tool for a certain product is lost for production. The total set-up time of a tool depends on the tool's utilization, set-up durations, the product mix, equipment dedication, and the dispatching rule.
- ▷ Batching policies. For batch tools the question arises when to start processing a new batch. Waiting until the maximum load

size of parts is available may cause delay in the overall cycle time. Starting the process with fewer parts reduces the tools capacity. Thus often certain thresholds T are set up: the machine is started when at least T parts are in the batch run.

- ▷ Dispatching policies. The influence of dispatching policies on the overall system capacity is due to the effects on batching and set-up. Moreover, dispatching might affect the system's variability.
- ▷ Hot lots/Engineering lots. Hot lots may increase the cycle time of regular lots by causing regular lots to wait, by forcing additional set-ups, forcing the processing of small batches at tools with large load sizes, and by increasing variability.
- ▷ Operator dependencies. Together with capacity planning the number of operators has to be determined. Even modern tools require operators for proper operation. When tools stay idle because no operator is available, time is actually lost if it is a bottleneck tool. As it is too costly to staff every single tool with an operator, operators must often handle several tools. In this case the skill of the operators working together in a shift and their level of cross training gets important.
- ▷ Rework. If the quality of a part leaving a tool is not sufficient it can be reworked either at the same tool, or follow a more complicated rework flow. This requires additional capacity as the tools for rework are often the same used in the main flow. Moreover, inserting parts back into the main flow increases variability.
- ▷ Product mix. The product mix determines the set-ups necessary and affects dispatching and batching strategies. Moreover, as different products are processed according to different recipes, the load on the tools may vary with varying product mix. Questions arise whether to qualify more tools for a product. With short product life cycles and increasingly complex products, i. e. an increasing number of operations, the product mix influences capacity considerations.
- ▷ Lot size. Often many different batch sizes can be found in a production line. This is because different tools may have different processing batch sizes. Moreover, the transport batch size might also be different from those processing batch sizes. Changing

Long range	Medium range	Short range
change country and/or facilities, change capital equipment, change work force	change make/buy decisions, plan alternate routings, subcontract (medium periods), re-allocate work force, change work force if feasible, add additional tooling	schedule overtime, subcontract (short periods), select alternate routings, re-allocate work force

Table 1.2.: Possibilities of adjusting capacity

between different batch sizes might cause parts to wait, increase set-ups, or increase the number of runs needed.

Depending on the planning level several standard capacity planning techniques can be distinguished in increasing order of detail [Smi89]:

1. *Capacity planning factors.* The distribution of orders between different manufacturing resources is solely based on historically determined capacity factors.
2. *Bill-of-capacity.* This technique extends the first method by taking the bill-of-materials and the product mix into consideration.
3. *Time-phased bill-of-capacity.* A dynamic method based on backward scheduling while considering the master production schedule (MPS), bill-of-materials, and work center information.
4. *Capacity requirements planning.* The capacity requirements planning extends the third method by also taking orders, on-hand inventory, and work-in-process inventories into consideration.

The capacity planner has got several options to adjust the production line's capacity to the capacity needed. The options — divided by the time horizon are shown in table 1.2. As will be shown in later chapters, *integrated simulation* includes detailed capacity planning and the terms defined here will also apply to queueing network analysis.

1.4. Modeling and Simulation

According to VDI guideline 3633 simulation means copying a system including its dynamic processes into a model that can be used for experiments in order to get to insights that can be applied to reality. In a broader sense simulation means to prepare, to carry out, and to analyze selected experiments with the help of a simulation model [VDI93, translated].

Many systems are that complex that there is no chance to describe their dynamic behavior analytically. In this case simulation is an appropriate means to study the system's behavior. Examples of such systems are production lines of semiconductor manufacturing with highly re-entrant lines. Typical questions are estimates of utilization, work-in-process, lead time, and throughput. Moreover, the influence of different dispatching and release policies is an important question.

The complex system to be studied is called *real system* or *real-world*. A simulation study involves the following steps [Möl92, p.84]:

1. *Qualification*. Qualification refers to the modeling process, i. e. to find the important elements, their attributes, and relationships in the real world and arranging them in an abstract model. Important is the choice of model. This can be simple equations, differential equations, queueing theory formula, discrete event models, etc.
2. *Rectification*. Rectification is the construction of a real model based on the abstract model. This can be a numerical method, implementation on a computer, matrices of a queueing theory analysis, or the construction of a model for discrete event simulation for a special simulator.
3. *Verification*. After a real model has been constructed it has to be verified. For different parameter sets the model has to provide results that correspond to the behavior of the real world. Only if this can be achieved, it is reasonable to use the model to predict results for parameter sets that cannot be evaluated in the real system. Succeeding in the prediction of results that can be observed in the real system is called *validation*, failing to do so is called *falsification*.

The interaction of these steps is shown in figure 1.4. A similar approach can be found in [Sch87].

In general there are two types of systems, discrete and continuous ones. A discrete system is one for which the state variables change instantaneously

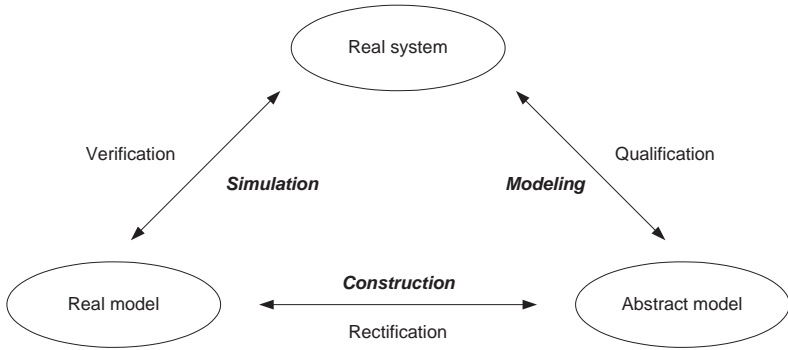


Figure 1.4.: Modeling and simulation [Möl92]

at separated points in time. A manufacturing system is a discrete system as the state variables, the number of parts at work centers, change only if processing a part has been finished. The suspension of a car driving through a road hole is a continuous system, since its state variables such as position, velocity, and acceleration can change continuously with respect to time. Its behavior can be modeled by differential equations. Few systems in practice are wholly discrete or wholly continuous, but since one type of change predominates for most systems, it is usually possible to classify a system as being either discrete or continuous [LK91].

In the remaining parts of this thesis the main point is to analyze complex production systems. Therefore now the two most important techniques to study such systems are presented, namely *discrete event simulation* and *queueing network analysis*. The main points of the analysis are queues of parts in the system. These are mainly caused by batching processes at work centers, transportation, and by variation in the manufacturing system due to random effects like irregular arrival of customers, machine breakdowns, varying service times, etc. Although enough capacity is available on the average, waiting parts/customers cannot be avoided. Thus, the goal of the analysis is to find sources of variation and to avoid them. Examples of production systems with largely removed causes of variation are continuous flow lines with equal cycle times at successive operations. Unfortunately, work centers in semiconductor manufacturing are very complex and different from each other with respect to cycle times and batch sizes. Consequently there

are hardly any flow lines in this kind of industry.

For the analysis with discrete event simulation the production system is modeled as a system of discrete objects that interact with each other in a well-defined way. Each interaction is called an *event*. These events happen at certain discrete times and are logged into an event list, which is simply a calendar of events due to happen. Each event is of a certain type and is processed chronologically, which may cause new events to be scheduled, until some termination criterion is satisfied. The details of discrete event simulation can be found in text books, like [LK91], for example. Further references can be found in [BN71], an early work on simulation, [NS93] describing applications of enterprise planning, [Jün89, p.577], simulation based planning, or [Pay88, p.225] that presents verification and validation procedures. A lot of authors consider discrete event simulation to be appropriate even for large production lines as in semiconductor manufacturing and report successful simulation studies in various real life settings, for example [HF99]. However, the most important drawback of discrete event simulation is that detailed simulations require a great deal of time and money to be set up and maintained and usually take at least several hours to run on even powerful computers. For a statistically significant analysis either long run-times are required to detect renewal cycles or a large number of repeated experiments has to be performed, confidence intervals have to be determined and if different scenarios have to be evaluated, multiple means procedures have to be applied [Har93]. Taking that much time it is hardly possible to perform many what/if scenarios or even optimizations. Moreover, many questions can be answered much faster with analytical methods. These include capacities, for example. To discover whether a production system has enough capacity to produce a desired demand, simulation may require a very long time, although questions concerning capacity as well as most other questions on the tactical planning level can be answered much faster with the help of the analytical techniques based on queueing theory.

If an analytical solution to a mathematical model is available and is computationally efficient, it is usually desirable to study the model in this way rather than via a simulation [LK91]. Of course, a queueing model of a large production line is far from being simple. It involves solving large systems of linear equations and complex approximation formulae for the performance measures. The details of queueing network analysis and the techniques realized by the EPOS simulation server are presented in section 1.4.

The pioneering work of queueing theory was done in the beginning of

the 20th century by A. K. Erlang, a Danish engineer, working in the field of telecommunication (see [BmJ60]). He started the analysis of queueing systems characterized by a Poisson input process, exponential holding time, and single service channel. Since then research has been focusing on more complex queueing systems like multiple servers, multiple classes of customers (products), general distributions of inter-arrival and processing times, batch processors, and priorities. Moreover, results from single work stations have been extended to yield results on networks of work centers. Most of the known solution approaches analyze the steady state behavior. The performance measures of interest are expected work-in-process (queue lengths), lead time, throughput, yield, and utilization. Analyzing the transient behavior of queueing networks is even more difficult and one of the main fields of current research.

In general, queueing networks can be divided into *open* and *closed* networks. In open networks service is unrestricted and an arbitrary number of customers might enter the system. Examples of such queueing systems are shops, petrol stations, toll bridges, non-Kanban production systems, and computer tasks to be scheduled on processors. In closed queueing networks the potential number of customers is restricted. These networks are a model for the repair man problem (a fixed number of workers servicing some tools), a machine operator in a cell with several tools, and Kanban production systems.

Despite their computational power queueing networks are hardly used in production planning. This is probably due to their mathematical complexity. In contrast to queueing network analysis, there are a lot of systems commercially available for discrete event simulation with user-friendly interfaces.

With these two methods, discrete event simulation and queueing theory, available, the question arises which one is appropriate to analyze the behavior of complex manufacturing systems. There cannot be a general answer, it depends on the questions that are to be answered.

The power of discrete event simulation is that transient behavior can be explored and that a manufacturing process can be modeled in great detail including batching and scheduling decisions and operator dependencies, for example. Most simulation systems provide standard components for these tasks, and what cannot be modeled directly can be programmed in most simulation environments. Of course, the construction and validation of such

models take much time as well as simulation and statistical analysis.

Queueing networks allow a very fast calculation of the steady state performance measures of a production system. The results of queueing formulae are expected values and variations. These are computed directly whereas discrete event simulation just produces samples that have to be evaluated statistically. Unfortunately, not all situations can be analyzed with analytical formulae. At the moment there are few results for transient behavior, priorities, and scheduling policies in queueing networks. Moreover, hybrid networks as a model for open manufacturing networks including closed networks for operator dependencies are not available. Thus there are limitations in the modeling power that cannot be overcome by programming new components as can be done in most discrete simulation systems.

As a general guideline, discrete event simulation should be used for operational planning, i. e. the analysis of release, dispatching, and scheduling strategies and for the detailed analysis of smaller cells of the manufacturing process. Queueing models show their strength in tactical planning, where work-in-process and lead time estimates are needed without modeling the very details of the manufacturing system.

Chapter 2

The Model of Integrated Simulation

Whereas the previous chapters showed the tasks and difficulties of production planning in modern production environments, this chapter presents the model of *integrated simulation* which serves as a foundation for an accurate and fast production planning system on the tactical level.

The previous chapter strengthened the need for elaborate methods that go beyond simple spreadsheet analysis. Discrete event simulation or queueing theory are the methods of choice when it comes to evaluate the performance of complex dynamic systems. However, these techniques require more detailed models of the production system in question. Those models require much time and data to create before the questions of production planning can be answered at an acceptable level of validation. Moreover, modern production environments are subject to frequent changes as shown in section 1.1. Thus an acceptable validation level must not only be reached, but also has to be maintained. This is a very difficult task as simulation models require a huge amount of parameters.

By defining the general requirements that an efficient system for production planning has to fulfill, the model of *integrated simulation* provides a theoretical background on which systems for production planning can be implemented. This includes various aspects from general system design over

proper choice of modeling and simulation techniques to sophisticated methods of information distribution such that business processes for production planning are well supported. The efficient implementation of a system for *integrated simulation* is made possible by emerging internet standards and middleware technologies which are needed means to connect various types of systems.

2.1. Analysis of the Current Situation

Today, there are mainly three different techniques for the evaluation of a production environment. These are

- steady-state spreadsheet analysis,
- single-user simulation systems,
- and mathematical models.

The planning tasks shown in section 1.3 have been well-known for several years by now. While sophisticated mathematical methods for production planning exist there is still the so-called practicality gap [How68]. Sophisticated mathematical models are often too difficult to understand, too difficult to fill with real-life parameters, and often do not apply directly to the problems of the real world. Thus these methods are either simply ignored by practitioners or only some basic results are used, i. e. the true power of such models is seldom exploited.

In practice, most of the planning is done on the basis of spreadsheets that offer much flexibility. Users can easily manipulate the data and formulae. However, this approach implies several disadvantages [RFN00].

The most important drawback of spreadsheets is that they only allow a steady-state analysis with rather limited formulae, i. e. dynamic behavior and random effects are very difficult to take into consideration. Moreover, spreadsheets are single-user tools. Sharing valuable planning data can only be accomplished by sending the spreadsheets to other users. Thus many copies of a spreadsheet are introduced. This redundancy is a reason for contradictory data, because if changes to the planning parameters are made, it is hardly possible to inform all previous addressees of the spreadsheet. Some of them might have changed the sheets themselves, thus making it even more complicated to merge the new data.

Often assumptions have to be made to make planning with spreadsheets tractable. For example, yield loss is modeled by a simple factor per product to reduce the output rate or additional load on tools. Rework is modeled by some additional start factors guessed. More sophisticated calculations could take those operations into consideration where scrap and rework occur. However, this requires more detailed data that is hard to obtain. But even if that data could be acquired, the task of solving systems of equations remains.

Moreover, capacity losses due to the reasons presented in section 1.3.3 are normally only accounted for by simple factors reducing the time a tool is available for production. The variation of these times is neglected completely. The same holds, for example, for guessed factors for down time and unavoidable idle times. They are simply modeled as contingency factors.

Furthermore, it is often difficult to get consistent data that is agreed upon to be used for production planning. Often planning managers have different ideas on tools utilization than manufacturing or engineering employees. There is no set of confirmed planning parameters, yet worse, sometimes the models that calculations are based on do not conform.

If planning data is taken from databases existing within the company, the planner faces himself with a huge variety of different, isolated systems of different vendors. There is no data warehouse that supplies the planner with a consistent data model covering all relevant aspects of the planning process that provides the data at the aggregation level the planner needs.

Integrating different data sources often leads to large consulting projects. But once a project of integrating different data sources and creating a simulation model is finished, it is open whether the model developed keeps on being updated. In environments subject to frequent changes permanent updates of the model are required.

Spreadsheets normally do not offer any means to evaluate the work-in-process or the lead-time because this requires more sophisticated methods. However, these are the essential performance measures that allow to set up a competitive production environment.

Moreover, most spreadsheets have difficulties in taking the time-dependent aspects of a production system into consideration. Many parameters like product mixes, yields, process plans, etc. vary over time. Including all different settings at certain points in time requires much data and computational effort. (More details about the integration of data in spreadsheets can be found in section 4.4.1.)

In contrast to spreadsheets, (discrete event) simulation systems provide the means to handle the random effects existent in production systems. However, these systems are hardly used and if they are used, they are mostly de-coupled from spread data and other data sources. This means that all parameters have to be copied into these systems and that two — or even more — systems are to be maintained. For example, most interactive simulation tools allow to create a process flow by operations in a graphical user interface. This data is stored in most companies in the databases of the shop-floor-control system that controls and tracks the movements of production parts. Thus, the flow constructed in the simulation environment is redundant. As long as simulation software stays isolated as a single user tool, the task of generating and maintaining a simulation model remains too tedious for a simulation analyst.

Discrete event simulation offers the advantages of an easy-to-understand model as with most simulators the important features of the real word can be modeled directly into the simulation system. The problem of discrete event simulation lies within the execution time and the output data analysis. The generation of events for large systems takes much time. Due to the stochastic nature of most production environments, the corresponding simulation runs require a very long execution time to achieve results within appropriate confidence intervals.

Another disadvantage is that simulation requires special know-how in the area of modeling and statistics. Normally, the creation of an initial simulation model is a project by external consultants and simulation analysts. The planning departments do not always use the simulation directly but need the simulation analyst for model creation, changes, and interpretation of the results.

Apart from these drawbacks of planning methods, computer science has evolved different techniques for dealing with distributed computing, efficient data handling techniques, sophisticated software development and deployment techniques, and CSCW (computer supported cooperative work). These offer great opportunities as will be shown in the following sections.

2.2. Features Required for Integrated Simulation

After having discussed the current situation this section describes the holistic concept of a system that is supposed to overcome the drawbacks of common

approaches. The model of *integrated simulation* describes the integration of simulation methods into the business processes typically found in production planning and control. It is based on ideas from the fields of

- data-warehouses,
- computer supported cooperative work (CSCW),
- network computing and middleware,
- modeling and simulation, and
- queueing theory.

The main focus of *integrated simulation* lies on the use of elaborate simulation and planning techniques for large manufacturing environments which can usually be found in semiconductor manufacturing. Simulation and/or queueing models for such production lines require a controlled process to assure that the necessary parameters are collected in an efficient, correct, and non-redundant way. A persistent model of this data has to be stored in a database. The data sources to fill this database have to be identified and interfaces have to be generated that allow to import the data from the company's databases into a single consistent data model used for all planning activities. In this context the central database can be seen as a data warehouse. Input to this data warehouse is derived from ERP systems, shop-floor-control systems, and often it cannot be avoided to allow for manual input as well. The planning data itself might be provided by employees from different departments. A controlled process allows a collaborative way of creating and maintaining simulation and computation models. This is needed as detailed models require more accurate input data than crude models. The details of the process of gathering the data for planning and simulation are described in section 2.4.2.

As all planning algorithms have to work on the central database, this is the most important step away from spreadsheet planning as spreadsheets require formulae to be specified redundantly, i. e. they apply a multiple data – multiple formulae model. Unfortunately, many users are accustomed to their spreadsheets and need the freedom of the data manipulation possibilities a spreadsheet offers. A good way to solve this problem is to let users work with their spreadsheets for easy calculations, but to import the planning data first and update the spreadsheets from the data warehouse. This

allows to benefit from the data warehouse once it is set up while still using the planning sheets the planners are accustomed to. But nevertheless, programs written in a programming language are usually faster (as they are compiled instead of being interpreted) and offer greater flexibility to the developer than spreadsheet formulae or script languages. Standard routines for handling complex data structures, like sparse matrices and solvers for systems of equations, are available as libraries for the C or C++ languages, for example, and thus are easy to incorporate.

The next step is to integrate the simulation and queueing methods. As the main drawback of single-user simulation systems is the long time it takes to create a model, this task has to be automated on the basis of the data collected in the data warehouse. This includes all steps a simulation analyst would normally carry out: generating sub-models for complex tools, creating a model for certain planning tasks, i.e. a parameterized model generation, for example. In any step the features of the simulator to be used have to be accounted for.

Integrated simulation is designed to be a flexible model that supports all planning tasks and simulation methods. Thus its data model has to consider the superset of all parameters needed for different simulators that should be used even if not every simulator or planning algorithm makes use of the whole set of parameters.

Integrated simulation requires a simulator that can be tightly integrated. The ability to import models is not enough as it must be possible to create models by other programs, to start and to simulate these models and export the simulation results; i.e. the simulator must be controllable via an application programming interface (API).

Moreover, the simulator has to be fast: This means steady-state results — the type of results which are needed on the tactical level — have to be returned in less than a minute. The reason: The simulation is carried out in the ongoing production planning process, not in a separate phase before the production line is constructed and one might be able to run the simulation over the weekend. When a new-built program arrives, the simulation report has to be ready a few hours later.

In general, it makes no difference whether the simulator is based on discrete event simulation or queueing theory as long as the right level of detail is supported. The simulator does not need to be able to model every artifact needed, as long as the automatic model generation can help to create

sub-models that will lead to an accurate modeling. It makes no sense to use a simulator which cannot model the aspects of real-world production lines, though. On the other hand, the simulator should not force the model to be too detailed. The right amount of detail is important. Right now queueing theory is more suited, but this is not a requirement.

Simulation always requires the validation of the model. This can be incorporated as well by checking the input and output data of the simulation against real-world data from a company's databases like work-in-process tracking systems, tool utilization monitoring systems, etc. A detailed discussion on integrated model validation can be found in section 2.8.

The most important part concerns the business processes. As seen in the last section, conventional simulation studies on single-user computers normally involve a simulation expert who serves as a means of communication between users and simulation. The expert asks for the parameters, constructs a model, simulates, and finally presents and explains the results to the users. In this loop of communication the simulation expert is the bottleneck.

Integrated simulation tries to avoid this permanent communication overhead by moving it to the design and implementation phases of the system. The goal is to integrate simulation results into the business processes of the company. In this scenario, the simulation expert helps to implement a system that delivers the answers to the users' questions. These answers are derived from the simulation results and are prepared according to the users' needs. Figure 2.1 shows the difference between the conventional approach and *integrated simulation*.

The results of the simulation have to be presented to the planners such that their questions will get answered without any additional effort. For example, planning processes that have to be supported are:

- Capacity, lead time and work-in-process planning
- Production line analysis: determination of work centers with highest utilization (bottlenecks), longest queues, and longest lead time, product mix analysis, etc.
- Yield management

These are described in detail in section 2.9. Adapting a server-based communication, not only the planner is able to analyze the results but also other

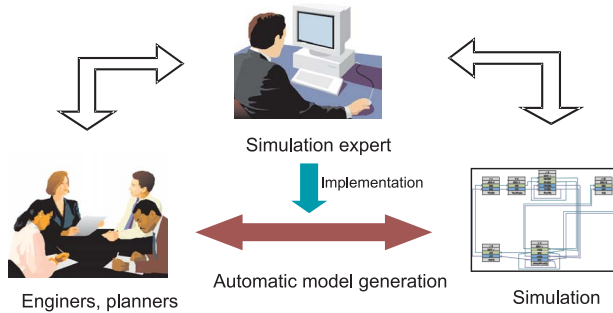


Figure 2.1.: Conventional simulation and *integrated simulation* (light and dark arrows, respectively)

employees that rely on the outcome of the planning processes; i. e. the planner should not have to prepare presentations and hold them once, but an intranet server offering the results allows anyone access to them whenever they are needed.

The summary of these ideas is shown in figure 2.2. From top to bottom it shows the three steps of *integrated simulation*, namely *distributed parameter input*, *automatic processing*, and the *distribution* of simulation results into the business processes of production planning (output). All these steps have to be supported by an IT system that integrates all planning and administration tasks.

With the improvements concerning IT infrastructure during the last years, it has become easier to develop a system that allows for realizing the ideas behind *integrated simulation*. One of these improvements is, for example, the wider use of intranets that clearly is a prerequisite for collaborative work. Moreover, databases allow to handle large amounts of data efficiently, and middleware technologies like JDBC or CORBA offer a rather abstract and thus convenient way to implement a system for production planning. From a technical point of view, three characteristics of a system for *integrated simulation* are essential:

- *Scalability*. The goal of the system is an integrated production planning including several production lines. This can lead to a huge number of simulation objects that have to be maintained. Those large models

2.2. FEATURES REQUIRED FOR INTEGRATED SIMULATION

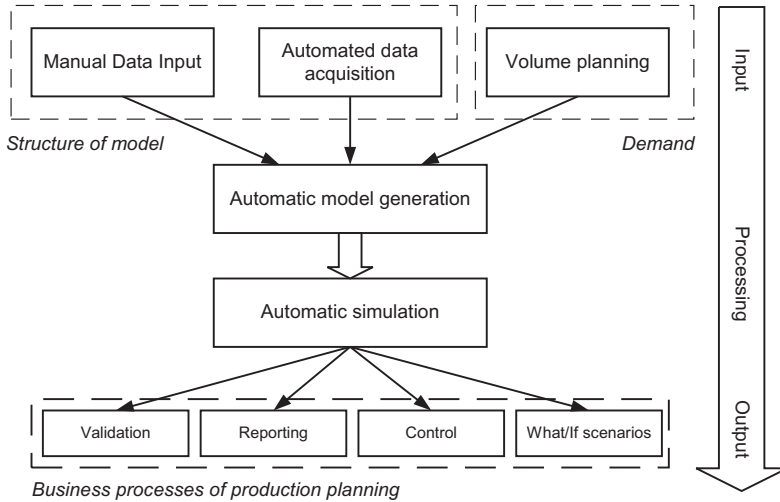


Figure 2.2.: Model of integrated simulation

have to be handled efficiently by the system, especially when for supply chain planning several of those lines are linked and evaluated together.

- *Openness.* The system has to offer standard interfaces that allow for other tools to access the master data as well as the results of simulations and computations.
- *Easy software distribution.* As a system for *integrated simulation* is supposed to enable the cooperation of a large number of employees with different roles and access rights, each user must be supplied with his own user front-end. In large companies, responsibilities and users change quite frequently. Thus the software has to be able to be used from anywhere within the intranet, without additional installation or customization requirements.

The requirements presented in this section lead to the conclusion that *integrated simulation* describes an e-business process supporting production planning. Simulation is integrated into the essential planning business processes with the help of modern IT technologies such that all steps in an

ongoing simulation project are well-defined, assigned to the employees responsible, and supported by a single, distributed system. A system for *integrated simulation* has to be permanently available for planning as required in [CIM91, p.63]. To be successful *integrated simulation* has to be the single system for planning. Other methods, like spreadsheets for example, have to be integrated such that redundancy and inconsistency are avoided. This requires an organized introduction of the system into a company and appropriate education because employees might be confronted with new techniques, tools, and methods.

To close this section the requirements for a system for *integrated simulation* are summarized based on the structure shown in figure 2.2:

- Data acquisition
 - ▷ transparent documentation of all parameters
 - ▷ a history of all relevant changes to the master data
 - ▷ archiving functionality
 - ▷ well-defined user roles for a cooperative, parallel model creation
 - ▷ ongoing validation
- Data processing
 - ▷ fast calculation of performance measures
 - ▷ support of what/if scenarios as well as automated evaluations
 - ▷ incorporation of modeling techniques
 - ▷ ability to parameterize model generation
 - ▷ ability to handle incomplete and inconsistent parameters (robustness)
 - ▷ possibility of using processing methods for educational purposes
- Distribution of results
 - ▷ permanent availability
 - ▷ hierarchically structured presentation of results
 - ▷ pull instead of push methods
 - ▷ access control (security)

2.3. Literature Review

Whereas many textbooks have been published on discrete event simulation and queueing theory during the last years, this review focuses on articles that have identified the need for more sophisticated support of simulation projects by information technology. Reference information on discrete event simulation and queueing network analysis can be found in section 1.4.

Many authors have recognized that the most difficult and tedious task to be performed in simulation projects is the acquisition of valid model data, for example Domaschke et al. [DRL98], Chance et al. [CRF97], Bey [Bey91, p. 54] and Trybula [Tyb95]. The latter measured the time spent on different tasks in several simulation projects. The results are shown in table 2.1. Normally these tasks are performed consecutively and not in parallel such that employing more people for simulation projects does not shorten the time necessary.

In [DRL98] the authors note the need for integrated systems, as the majority of the effort for most simulation projects is expended in collecting and preparing input data to construct a valid model of the factory. The analyst is supposed to spend his time on simulation analysis instead of data management. Thus the company for which the project experience is described in the paper set up a data warehouse which is filled by the IT and CIM departments with data from the operational systems. Thus the responsibility of data management is moved away from the analyst. The principle is that the centralized data warehouse is used for all modeling — static and dynamic — from simple spreadsheets over capacity planning to discrete event simulation

Tasks	Amount of time spent
Problem definition	~ 10%
Problem analysis	~ 10%
Data gathering and validation	~ 10% to 40%
Model development	~ 10% to 40%
Model verification and validation	~ 10%
Model experiments	~ 10% to 20%
Analysis of results	~ 10%
Conclusion and recommendations	~ 5%

Table 2.1.: Amount of time spent on particular simulation tasks [Tyb95]

analysis. The key premise is that all data used for decision-making models should come from the same data warehouse.

Furthermore, the authors describe some scenarios and parameters that can be changed in order to improve the overall factory performance. The studies briefly discussed in the paper include changing the test procedures, changing batching policies at an oven, tool dedication, staffing, lot release policies, transportation lot sizes, and operator scheduling. Note that although the authors include different batching decisions as decision variables the sample operating curve (line profile) for the overall cycle time does not show the typical bath tub curve. Moreover, the paper does not mention the model size or the time it takes to execute the analysis.

Another approach based on a database can be found in [CS93]. The authors describe an approach to create a simulation modeling environment (SME) around a relational database. Their goal is to provide standardized information handling and to realize the concept of language neutrality: the ideal SME should allow to create, edit, and store systems from which models can be defined. Moreover, the authors appreciate the possibilities of a proper data model, the structured query language SQL and of several users working in parallel with specified user interfaces. In this setting simulation traces and results are stored in the database as well. The focus of this article lies more on the modeling environment, not that much on the industrial application.

[CRF97] as well point out the difficulty in obtaining proper data. They put special emphasis on the robustness of simulation models, i.e. the models have to be flexible enough to be changed over time by different people. Having rapidly changing market situations and technological trends in mind they put special emphasis on the dynamic nature of a simulation model. A simulation model should not be just valid at a certain moment in time, but it is to be validated and used permanently on an ongoing basis. The authors warn against inaccurate data that is used through oversight and they ask for clear lines of responsibility for collecting the data.

The need for an automatic model generation has been realized by Srinivasan and Jayaraman [SJ97]. Their motivation is the danger of inconsistency and the avoidance of tedious replication work. They also mention the automatic generation of SIMAN¹ code for simulation. In contrast to the automatic model generation they assume the enterprise data to be consistent in order to generate simulation models.

¹a system for discrete event simulation

Further ideas of integrating simulation environments and data management technologies can be found in [WM93], [KRW93], and [Pfo92]. The last author discusses the modeling of logistic information systems and quality management in logistic information systems (p.163). Another author who emphasizes the quality of the data used for simulation and planning is Kuhn [KWB92]. He defines the quality of data as the efficient provision of the correct information at the right time, in the right place, in the correct state and amount. All these authors point out that any kind of planning is heavily dependent on the data used and agree that much effort has to be put into acquisition and administration of the master data.

In [HF99] the authors describe the ideas for an integrated and automatic model building process. The authors point out the importance of automated processes because otherwise simulation models are not maintainable and users lose confidence in the simulation results. The process described started out with a base model, including equipment data, product data, guided vehicles, and material stockers. Once the first model was built, the initial validation was done by collecting historical factory data from databases and comparing it to the simulation output. The data collected from factory databases included equipment data (work-in-process, throughput, utilization, cycle time), product data (work-in-process, yield, on time delivery, cycle time), and material movement data (bay to bay delivery times, stocker inventories). Due to the frequent changes in the demands and the production environment, the authors note the need for automatic model generation and validation. The simulation model is evaluated by a discrete event simulator, "[...] because it could evaluate the interaction between product mixes, taking into account time dependencies from recursive flow, and the stochastic nature of events in manufacturing." The paper does not say anything about the size of the generated models and the execution times. A future project is the establishment of a robust data warehouse for production planning and simulation. This allows the use of the true functionality of a data-mining tool integrated with reporting and modeling tools based on a proper data model.

As will be shown in later chapters, EPOS is such a data warehouse that integrates all planning data and that allows the use of report generators and data miners. Furthermore, *integrated simulation* extends the approach because it is not just a parameterization of a model, but also incorporates complex modeling steps, like sub-models for cluster tools, for example, and a

parameterized model generation that allows to generate automatically simulation models for different planning tasks.

2.4. Collaborative Creation of Simulation Models

As stated in the literature review in section 2.3 and in the model of *integrated simulation* in section 2.2 most simulation studies require a great amount of data. To obtain this data the model requires to collect the data at the source of information, which are either factory databases or the responsible employees. This leads to a collaborative creation of the simulation models and requires a process that controls this collaboration. In order to describe this process certain tasks and roles in responsibility of these tasks have to be defined.

2.4.1. User Roles

The notion of user roles is often used to describe the responsibility for certain tasks needed to accomplish an overall goal (see [Bur97]). Figure 2.3 which is based on Jablonski [JBS97] formalizes this concept. A role can be characterized by tasks which it defines and the resources that it has access to. Users act in roles which are assigned and administered by the organization to which the users belong. To successfully and effectively execute a role which requires certain skills users should obviously have acquired these skills.

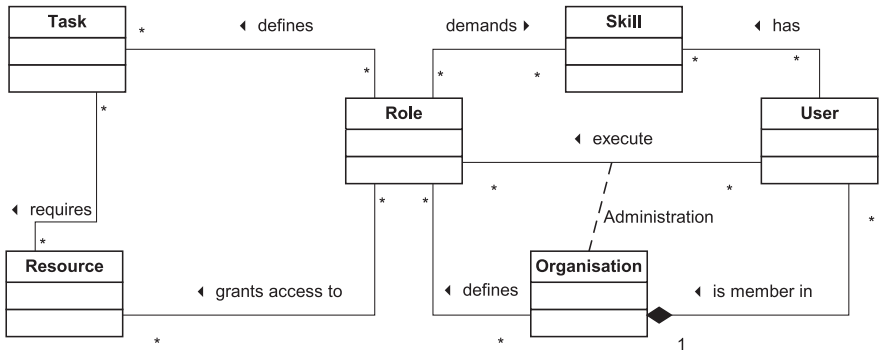


Figure 2.3.: Abstract concept of user roles

In the course of this section the abstract model of roles, associated tasks, skills, etc. will be substantiated with concrete definitions of roles and the interaction occurring to accomplish the overall task of achieving and maintaining a high level of model validation.

Overview

Production planning with *integrated simulation* requires support on different levels within the company. The senior management has to support the introduction of the system as the implementation of a system for *integrated simulation* is a rather large project requiring some resources. The benefits can only be gained if the system is integrated tightly, i. e. if the parameters for the simulation model are not kept up-to-date the results of the simulation will turn useless. Support is also needed by users who do not directly profit from the effort demanded by the system, e. g. the engineering department bears the load of maintaining the largest part of the data needed, thus facilitating the work of the production planning department. It is once again the management's task to clarify this situation and motivate users in order to counteract this inherent asymmetry.

Logical User Roles

Integrated simulation requires employees on both levels, on the technical and on the logical level. On each level different user roles can be identified demanding different requirements with respect to tasks, skills, resources, etc. Tasks on the logical level deal with the simulation model and its parameters itself. The following roles are defined:

- *Parameter supplier.* As *integrated simulation* tries to gather parameters for a simulation model directly at the source of the information, this is the largest group of people involved. They stay on the real-world level, i. e. they do not need to bother about details of the simulation. Provided with a sophisticated, secure, and user-friendly front-end, they can directly enter their parameters for work centers and process steps. The assignment of responsibilities assures that only the entitled persons can modify the data. Typically, the employees providing the parameters can be found in the engineering, manufacturing, and maintenance departments.

- *Department manager.* The managers of the parameter suppliers have to approve the parameters entered into the system. This is to assure that only parameters that have been checked by several people are used for simulation and that in case of wrong decisions caused by wrong parameters the responsible manager can be identified. Approval could be done by electronic signatures, for example. A manager signs the parameter set for a work center. If he is not content with some values he has the possibility of rejecting values entered. In this case a discussion starts until the employees involved agree upon a set of parameters for simulation. Approval or rejection should be possible by just clicking on an appropriate button.
- *Production line administrator.* The production line administrator has to care for the production lines master records. This includes
 - ▷ Products. New products have to be created in the data warehouse before the parameter suppliers can enter the data. When a new product is to be introduced the production line administrator has to decide which parameters of an existing product can be copied to the new one in order to keep the effort of the parameter suppliers at minimum. Moreover, the administrator decides whether and how products are aggregated to product families.
 - ▷ Work centers. The administrator has to decide how many and which work centers are to be included for simulation, i. e. to decide if there are several identical tools, whether they can be grouped into a single work center or whether each of the identical tools is dedicated to a special product. In the latter case, separate work centers should be created.
 - ▷ Operations and process flow. The administrator creates the operations for which the suppliers have to enter the cycle times etc. This might involve decisions about splitting complex real-world operations if the simulation model requires a higher granularity. In addition, process flows of current product groups have to be administered. This includes the definition of scrap and rework rates at different stages of the production process.
 - ▷ Interface to shop-floor-control systems (SFC): The administrator defines the mapping between entities in the SFC system and the

planning data warehouse. Moreover, he is responsible for the import of external data.

All these administration tasks require a thorough understanding of the data model of *integrated simulation*. The production line administrator must also be informed about the current status of the production line. Having to communicate with different departments, he or she is the mediator between the different users of the system, somebody who feels responsible for the production planning process. The role therefore also requires basic skills in this field, i. e. the administrator should be able to interpret simulation results and present these in management meetings.

- *Capacity planner.* The capacity planner is the main beneficiary of the *integrated simulation*. Simulation runs have to be started and the results have to be discussed with the production management. This includes the explanation of modeling assumption and the quality of the input parameters.
- *Volume planner.* The volume planner is responsible for the administration of volume plan versions. New versions have to be inserted into the data warehouse and current versions have to be assigned to production lines. It is the task of the volume planner to negotiate between capacity planner and the marketing department. He or she is responsible for accepting and committing the volume plan.
- *Simulation expert.* The simulation expert is supposed to support the production line administrators and the development team in their daily tasks if problems arise. Concerning the administrators, this includes questions on how to model certain tools, when and how to decompose certain operations, etc. He provides the development team with latest information on simulation and modeling techniques.
- *Consultant.* The deployment of a system for *integrated simulation* is likely to be a rather large project which often cannot be handled by users involved in the production planning. Consultants can help to deploy the system and customize features to fulfill the customer's needs (compare [KT97]).

- *Developers.* With the rapid pace of changes in the production environment the need for adjustment of software components and development of new features and interfaces arises. This is done by developers who need skills in the used software systems. Some — but not all — development activities also require knowledge in production planning and simulation methodologies.

The roles presented here define tasks for *integrated simulation*, they do not necessarily require full-time employees to fulfill the tasks. Furthermore, one person can take over several roles, for example, a production line administrator could also care for volume plans and start simulations.

Technical Roles

The previous section 2.4.1 described several user roles on the logical level. In addition to those, roles that define tasks involving the administration of software artifacts needed in a system for *integrated simulation* need to be described.

The efforts for the technical users have to be distinguished between the first time installation and the ongoing system support. Whereas the first is a complex project, the ongoing support is just monitoring a couple of servers. The roles are:

- *Database administrator.* Simulation models contain huge amounts of structured data. Independent of actual implementation, the best way to maintain this data is a database which has to be administered.
- *Administrator of the simulator.* The core of a system for *integrated simulation* will be a simulator that carries out the simulation. This is normally a rather complex software system which has to be set up and maintained.
- *Administration of reporting system.* To integrate results of the simulation into the business processes of the production planning some kind of on-line reporting system is needed to distribute the information to the users. This involves the administration of a security system as simulation results present confidential data which should only be made available to authorized persons.

- *Network and system administration.* Through the integration of many users, parameter suppliers, for example, *integrated simulation* heavily relies on the existence of a properly working network. This has to be maintained, as well, as the whole system environment.

2.4.2. Distributed Parameter Input

Simulation models for large production systems require a lot of parameters. For example, work centers have a mean time between failures (MTBF), a mean time to repair (MTTR) and a number of tools. Products need their specific production flow and demand². What makes the whole process of simulation even more difficult is that these parameters are changing over time. As shown in the introduction, the production environment is continuously changing. Parameters that are likely to vary in the course of time are

- *Demand.* The demand varies according to the customers' orders. Production planning is usually based on forecasts and consequently forecasted demands are different for each period.
- *Number of tools.* Varying demands and an increasing complexity of the products ask for changes in the production capacity. New tools have to be bought and older ones are removed from the production line.
- *Product flow and recipes.* New products or enhancements lead to updated flows with new recipes.
- *Rework and scrap rates.* When a new product is produced for the first couple of periods, normally the yield is lower than in the following periods (learning curves).
- *Production calendar.* Usually expensive modern plants are run on 24 hours a day, seven days a week. But there might be exceptions like weeks around Christmas or Easter and certain weekends at which production is stopped for maintenance. These additional down times have to be considered as well.

The following sections describe the potential sources of parameters and processes to incorporate the information.

²In section 4.4 all necessary parameters are described in detail.

Shop-Floor-Control Systems

In order to avoid redundancy as much information as possible should be gained from shop-floor-control systems. The following data is usually contained in shop-floor-control systems:

- Process flow
- Historic scrap and rework rates (overall or by operation)
- Current work-in-process
- Tool utilization data

The existence of this data does not guarantee that it can be easily incorporated into simulation models. Prior to a successful simulation some other problems have to be solved:

- *Integration of SFC data in the data model of planning data warehouse.* If the shop-floor-control system uses any foreign keys at all, they are likely to be different from the ones in the planning database. Thus a proper mapping has to be defined and maintained.
- *Different aggregation levels.* Often, SFC systems work on a different level of detail compared to the planning system. This can concern several items:
 - ▷ Products. It might be sufficient to consider product families or ignore the latest process change for a certain product. An aggregation level has to be defined for the input of the product data. This level should be rather detailed as further aggregations can be carried out later in the planning data warehouse or during the model generation for simulation studies.
 - ▷ Work centers. The SFC system might use different tool groups than the simulation system, due to other selection criteria.
 - ▷ Scrap and rework rates. These are random values; the rates have to be estimated from the transactions in the work-in-process tracking system. The problem is to decide what time interval is to be considered. The shorter the interval, the more topical is the data (learning curve especially for rework and scrap). However, the sample size is reduced, which leads to stochastic errors.

- *Errors in data.* Automatically collected data always bears the danger of containing random or systematic errors, especially if the data has never been used before for other purposes.
- *Historical data not needed.* SFC systems often contain historical information that is not needed by current simulation studies. The problem is to distinguish between the obsolete and the important data. For example, it is unacceptable to find obsolete products in the SFC and asking employees to enter data for it.
- *Several production lines.* If several production lines for the same products and processes exist, then a decision has to be made whether, for example, the operation objects are used for both simulation models or not. The general question is which entities are identical and which are equal but have to be copied for different production lines.
- *Frequent updates.* Another problem are frequent updates in the operational systems. If, for example, new flows are entered into the system the operational system might be temporarily in an inconsistent or faulty state. Frequent delete, edit, and insert operations make it difficult to keep the planning data warehouse up-to-date.

ERP Systems

Enterprise resource planning systems normally have a wider scope than shop-floor control systems. Among other things, they are used for planning orders, the explosion of bill-of-materials etc.

Especially the ERP system R3 by SAP [KT97] introduced the interface *Production Optimization Interface* (POI) that allows the export of ERP data for use in external simulation and/or optimization programs. Moreover, it is possible to upload the results calculated in order to make use of them later. The master and transaction data that can be exported includes [KC00]:

- *Master data.* Materials, bill-of-materials, work centers, work center hierarchies, process plans, resource networks, production/factory calendar, classifications, product groups, routing matrix
- *Transaction data.* Planned orders, production/process orders, stock/-requirements list, run schedule headers

- *Financial data.*

As long as ERP systems do not integrate simulation modules properly, they will not provide all data needed for simulation runs of future scenarios. Thus the need for additional information maintains. A system for *integrated simulation* serves as a means of providing this data.

Even if all data needed is part of the ERP system the problem of distributed data input remains. Allowing many people to access the data requires many user licenses for the system. This often results in high additional costs. Moreover, each user needs a user interface for his input. This increases the costs for licenses and software roll-out as well. In contrast, the model of *integrated simulation* requires an easy and efficient software roll-out as shown in section 2.2.

Process for Gathering Additional Information

Unfortunately, not all parameters can be taken from shop-floor-control systems as these systems are normally designed for control and tracking rather than for planning purposes. This results in data models that do not support the planning tasks so that often not all parameters needed are recorded. Moreover, as simulation studies are supposed to analyze future scenarios, parameters of new products, recipes and tools have to be entered manually. Other time dependent parameters, like demands, future rework and scrap rates, are often missing as well. ERP systems, on the other hand, often do not model a production line in the granularity required by simulation methods.

Parameters that cannot be imported from the company's databases have to be gathered and maintained in a special business process directly from the responsible employees. Many users should be able to input parameters directly and simultaneously into the system. The principle behind this idea is:

All data should be collected at the source of information.

This is necessary as for huge simulation models the simulation expert cannot efficiently collect all necessary data himself. Thus, the model of *integrated simulation* requires a process for data acquisition which keeps the administrative effort at a minimum and which reflects dependencies between the data input of different departments, access rights, responsibilities, and allows

for signatures by managers to yield approved data for simulation. All plan parameters are to be stored in a central data warehouse so that all planning applications can access this non-redundant and consistent parameter set.

These requirements can be realized in a distributed system. Today's IT infrastructure offers efficient possibilities for the realization of such systems, like an intranet, databases, and operating-system independent programming languages.

The process to gather additional parameters is founded on the following assumptions: The simulation model is constructed including n different data sets. Each data set contains the parameters one department is responsible for. The employees of that department enter the information needed. Once they have finished the managers of the departments involved have to look over the data entered and they either approve or reject it. Thus, *integrated simulation* introduces a 2-stage process for parameter input and approval. The two stages are:

1. *Data input by employees.* The parameters have to be collected at the source of information, i. e. directly from the responsible employees.
2. *Parameter approval or rejection by managers.* The responsible managers have to approve the data input. If they are not content with the parameters specified, the data has to be rejected.

The central objects for which parameters are to be collected are work centers as these represent real-world objects for which persons can be made responsible for. Parameters of a work center can be classified into different groups. In general, at least the four groups *maintenance*, *manufacturing*, *engineering*, and *staffing* can be identified. Thus, responsibilities for maintaining parameters can be split among work centers and also types of parameters.

As soon as the base data (products, operations, etc) is set up the employees can start to enter their parameters into the data warehouse. Once all parameter sets are defined and approved the simulation model is based on approved data. Of course, this does not substitute a proper validation of the simulation model, but it is a first step towards a valid model

As described in section 1.1 the production environment changes frequently. *Integrated simulation* considers this in two ways: It is always possible for employees to modify the data they entered. In this case the signatures of the corresponding manager has to be removed.

Although possible, it is rather unlikely that the employees update their parameters voluntarily. Thus sometimes it is necessary to force the employees to look over their data again by removing all signatures and automatically invalidating all parameters. As the parameter sets are well-defined, it is possible to control the process of approval.

The process presented implies several technical requirements:

- *Easy distribution of front-ends.* As a large number of employees is supposed to input data for simulation, the distribution of the software for the user front-ends has to be simple and effortless.
- *Easy to use front-end.* The user interfaces have to be adopted to the language of the employees. All references to simulation or queueing theory have to be omitted, as the users providing the parameters should not be bothered with tedious modeling details.
- *Additional attributes for specific work centers.* The front-end needs some possibility of demanding additional parameters for certain types of work centers. This implies that the GUI has to be dynamically adjusted during program execution in order to ask for the additional parameters.

2.5. Automatic Model Generation and Transformation

The handling of large simulation models requires three major components, a persistence model, sophisticated user front-ends, and an elaborate application logic. Common client/server systems are two-tiered: a database server is used for storing persistent data and the front-ends allow for easy manipulation of that data. Thus these applications are two-tier applications.

Integrated simulation requires an additional third tier: A model generator is needed to create simulation models (for different simulators) and control the simulation. The key features which have to be provided are:

- *Modeling techniques.* Complex tools require mapping to sub-models with respect to the capabilities of the simulator (see section 2.7.1).
- *Parameterized model generation.* Simulation models for different planning tasks depending on various parameters need to be generated, see section 2.7.2.

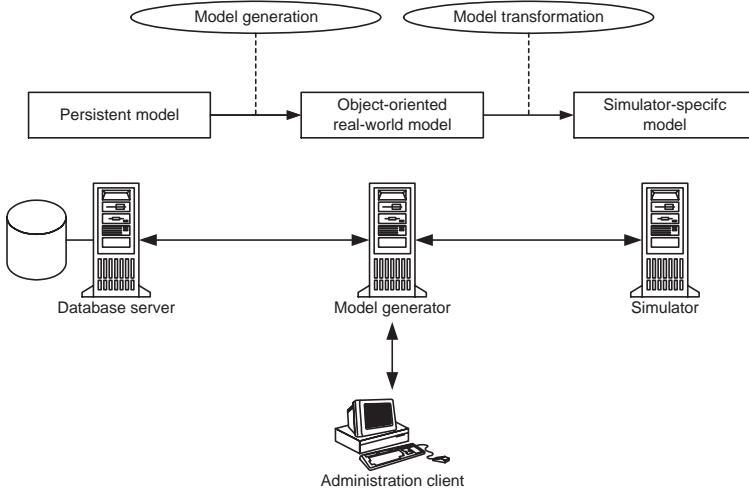


Figure 2.4.: Multi-tier architecture of integrated simulation

- *Handling of incorrect and missing data.* An application based on automatic generated data and manual user input depends on a complete set of proper parameters. If these are not available special actions have to be taken (see section 2.7.3).

The model generator demands a simulation server (simulator) for the execution of simulations. The interactions of the systems are shown in figure 2.4. Multi-tiered applications require a means of communication, the middleware.

The central data warehouse containing data imported from SFC systems and additional manual input provides the necessary input to generate simulation models for several planning purposes.

If a simulation client requests a simulation, the model generator generates the simulation model: The first step of the automatic model generation is the construction of the object-oriented *real-world model*. This represents the model stored in the data warehouse and is independent of any simulator or simulation language. Moreover, it contains the possible parameters for the automated parameterized model generation.

The next step is the transformation of the real-world model into a model for a special simulator or simulation language. The real-world model is not simulated itself, but it serves as a reference model which is transformed into models that can actually be simulated. The reasons for this are as follows:

- *Use of existing systems.* There are already various fast and elaborate simulators available. It is easier, more time-saving, and more error-prone to use existing, tested systems than to re-invent a simulator from scratch (see [Str92], [Boo94], [Bro95], [PS94]).
- *No simulator-related limitation.* The real-world model is not bound to any limitation that a specific simulator-dependent model might have. It must just be possible to transform the real-world model into the simulator model by the use of some algorithms. The limitations of the simulator used could be circumvented by the use of auxiliary constructs which can be created automatically, if possible.
- *Simulator independence.* An automatic model transformation yields the possibility of using different simulators. The strengths of different simulators and simulation methods can be efficiently leveraged just by supplying another transformation method.
- *Complexity.* A simulator is quite a complex software artifact. The real-world model tries to capture planning aspects of the whole enterprise. It is more complex than the simulation model in the way that only parts of the model need to be simulated together. This is important when it comes to maintaining the software system. The goal is to have modules with low coupling and strong cohesion (see [PS94], [Som96], [Boo94]). It is therefore reasonable to separate the simulation methods from the administration of the company-wide data. This reduces the complexity in each module without having to increase the coupling between them (see the interface definition in [Kle00a]).
- *Performance.* Performance is a very crucial factor in simulation (see [KW00] [GP00], [Arn00]). The implementation of a simulator will therefore strongly focus on an efficient program execution and a scalable environment. For the model generation which only takes up a small part of the total time needed for the simulation the program execution is not of first priority. Stability and interoperability are more

important in that area. Implementing and deploying the model generation and the simulation in two different environments are the logical consequence.

- *Integrity.* A simulation model must comply with certain integrity requirements [Kle00a]. The real-world model might not directly conform to these requirements as the process of parameter collection will very likely deliver faulty and inconsistent data (see section 2.4.2). A mechanism which *repairs* the real-world model so that it complies with the strict integrity rules of a simulator is needed.

The model of *integrated simulation* differs from other model generators by its two-step approach. It is not just setting the parameters of a simulation model by database lookups, but it includes various modeling steps hidden to the end-user. The generation and transformation of the simulation models have to be done automatically due to several reasons:

- *Model size.* *Integrated simulation* is supposed to handle models of complete production lines, i.e. the simulation model can consist of thousands of process steps.
- *Efficiency.* Production planners should not spend their time on tedious tasks which could be automated. Time won during automatic model creation can be used for the evaluation of further scenarios.
- *Easy data handling.* Updating simulation parameters can be quite easily done in a database, changes require the generation of a new model.
- *Parameterized generation of scenarios.* The evaluation of different scenarios requires several models. In large models it is hardly possible to generate complex scenarios manually.
- *Reproducible generation.* If a simulation model is damaged or becomes inconsistent, it can easily be generated from the data warehouse.

2.6. Generation of the Real-World Model

The first step towards a simulation model is the generation of the real-world model. This is an object-oriented model representing the production line.

The data imported from shop-floor-control systems and the manual input of the employees are taken from the data warehouse to build the object-oriented description of the real world. This requires the data warehouse to contain at least the data necessary for the creation of the simulation model. However, the real-world model might contain more information than necessary for a certain analysis:

- *Simulator independence.* Special features required for a certain simulator might not be needed in another one, e. g. dispatching rules for a discrete-event simulator might not be applicable in a queueing network. Nevertheless, a real-world model could contain this information.
- *Special analysis.* Certain what-if scenarios might neglect information, e. g. generating a scenario in which all tools work at their maximum load size ignores average or minimum load sizes.
- *Additional data.* The factory model is to be used for all kinds of planning tasks, even if they do not require simulation. Integrating all information into one consistent model ensures integrity. If the real-world model is hosted on an application server, client planning tasks can use the information provided.

The real-world model includes the main objects of the production environment that are necessary for the planning tasks. These are:

- *Production lines.* Production lines are the main entities on which simulation is based. Simulation models are always generated for a production line. A production line contains several cells, that allow to divide the set of work centers into logical units.
- *Work centers.* A work center is a group of several *identical* machines. Identity refers to the technical characteristics (like mean down time or mean time between failures) and to the use of the tool. If two technically identical tools are used for different products, i. e. they are dedicated, the two tools cannot be grouped into the same work center.
- *Products.* The class of goods to manufacture are the products.
- *Operations.* A product requires several operations (processing steps) on the way from its raw materials to the finished good. Each operation normally describes a step in this value-add chain.

- *Routings*. The routings describe the process flows of the products on the shop floor.
- *Volume plans*. The volume plans describe how many products are to be produced in a certain period.

A detailed description of the objects involved in the real-world model and their attributes can be found in the description of EPOS in section 4.2.

2.7. Transformation of the Real-World Model

After the real-world model has been generated it has to be transformed into a model specific to a certain simulator. In order to achieve this, the model generator has to establish communication links with the simulator of the transformed model. This can be done in several ways:

- *Use API of simulation tool*. If the simulation tool offers an application programming interface (API) this can be used. The underlying communication can be established by several means: libraries that are linked to the model generator, COM objects, remote method invocation (RMI), or the common object broker request architecture (CORBA). In general, all means of communication of distributed systems could be applied here. The method of choice depends on the capabilities of the simulation tool.
- *Generate model description for a special simulator*. A more loosely coupled construction is the exchange via files containing the model description. This might be ASCII files, for example, or other formats that can be imported into the simulation tool. In this case, the control of the simulation tool is more difficult. External batch programs are needed to generate the model and to start repetitive simulations. Moreover, these control programs are very likely to be operating-system dependent. Simulators are, for example, INSIGHT, Simple++(emPlant), Witness, MODSIM2, SIM++.
- *Generate simulation program for simulation languages*. Some simulation tools or languages, require input in the form of programs that need to be compiled, e. g. SIMAN, SLAM II, GPSS/H, SIMSCRIPT II.5, for

discrete event simulation. Basically, this suffers from the same disadvantages as the model exchange via files: Files have to be shared, and the control of the model generator and the simulator is not integrated.

Separating the simulator from the model generator requires a means of communication between those two. The communication requirements for a system for *integrated simulation* are:

- *Efficiency.* *Integrated simulation* is designed to handle large simulation models. Thus the communication between the model generator and the simulator should not be the bottleneck. Creating thousands of objects with the appropriate attributes and reading the performance measures for different aggregation levels lead to much data to be transferred.
- *Close links.* As *integrated simulation* aims at automatic creation of models control of every step of the model creation is useful, i. e. when parameters are set via the API of a simulator, the checking of the return codes of the corresponding methods allows for an early error detection. Simulators that apply checks when the whole model is created (which occurs if models are generated using files or simulation languages) are likely to waste time.
- *Transparency.* As standard simulation tasks are to be scheduled and carried out automatically, the activities have to be logged and the logs have to be distributed. Thus it is possible to control the simulation process from any environment.

As described above the transformation of the real-world model to the model simulated includes more sophisticated modeling techniques because in certain cases the real world cannot be directly mapped to a simulation model. For example, the simulation expert has to introduce new auxiliary work centers to model the behavior of complex work centers correctly. However, all aspects of complex model generation and modeling techniques involved in this step are to be hidden from the end-user as far as those aspects are not necessary to understand and interpret the simulation results.

Moreover, time-dependent characteristics of the simulation model can be taken into account in the parameterized model generation: Simulation models for tactical planning need to include future planning parameters, like the number of tools in a work center in the future, or a time-dependent yield.

If the client requests a model for a certain point in time, all time-dependent parameters have to be adjusted. This is part of the model transformation as well.

2.7.1. Modeling Techniques

The most important task of the model transformation is the application of the modeling step, i.e. in general there is no one to one correspondence between the work centers of the real-world model and those of the model specific to a simulator.

For these special modeling steps additional parameters are necessary: The parameters can either be an attribute of

- work center itself,
- an operation at the work center,
- a product at the work center,
- or any combination of the above.

Moreover, the performance measures of sub-models have to be aggregated so that the performance measures of the corresponding work center in the real-world model can be provided to the end-user. Both, the parameters needed and the calculation of the performance measures are different for each work center type. Typical examples of work centers requiring special modeling techniques are

- sequence tools,
- inspections,
- tools with subsequent process steps without capacity constraints, and
- cluster tools.

Sequence tools are similar to small flow-lines. A part is started and once it has finished the first step in the tool and is passed on to the second step, another part can be started. Thus successive processing steps can be run in parallel (*pipelining*), although no queueing is permitted between the process steps within the tool. Typically one of the operations performed in the tool

takes the longest time. This processing step is the bottleneck of the tool and determines the throughput.

Sequence tools can be modeled by two work centers: The first takes the bottleneck processing step and serves as a trigger for the second. The second work center is of infinite capacity and carries out an operation that takes the remaining processing time, i. e. the complete processing time minus the time of the bottleneck step. The trigger and the remaining processing time depend on the operation and on the product. Typical tools following this principle are cleaners in which the parts are put in different baths successively.

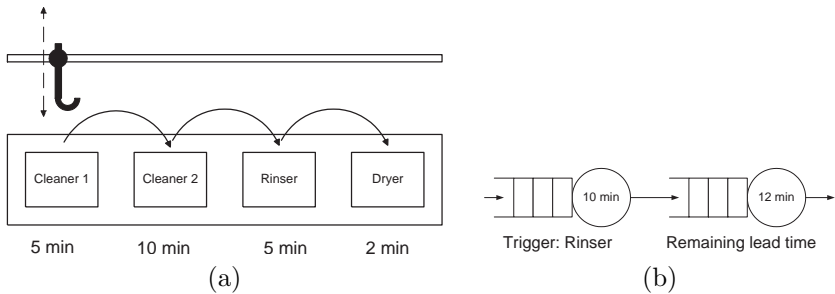


Figure 2.5.: Transformation of sequence work center

The performance measures of the sequence work center can be computed as follows:

- *Utilization.* The utilization of the sequence tool is equal to the utilization of the trigger work center. Note that the work center for the remaining processing time is an infinite server. Thus its utilization is zero.
- *Work-in-process.* The work-in-process of the sequence work center is the sum of the two work-in-process inventories at the trigger and the station for the remaining processing time.
- *Lead time.* The lead time of the sequence work center is the sum of the lead times of the trigger work center and the remaining process time work center.

Tools at which inspections are performed can be distinguished by their sample rate. The latter normally depends on the work center, on the product, and on the operation: Newer products need more testing, i.e. their sample rate is usually higher than that of products being produced for a longer time. Longer periods of production lead to more experienced staff and better processes. Thus the longer a product is produced the better is its quality (learning curve).

There are several ways to model a sample rate: (i) reduction of the cycle time by multiplying it with the sample rate for each process step and (ii) construction of a routing with a routing probability of 1-sample rate leading around the inspection operation. The first approach is to find out if parts not being inspected wait for others until the complete batch is moved on to the next process step. The second approach is appropriate if parts not being inspected do not wait for others to be finished. In this case they can be directly started at the next operation.

The performance measures of the inspection work center can be computed as follows:

- *Utilization.* The utilization of the inspection work center equals the utilization of the single work center constructed for simulation.
- *Work-in-process.* The work-in-process of the inspection work center is the work-in-process inventory of the single work center representing the real-world tool for both ways of modeling.
- *Lead time.* The lead time of the inspection work center is equal to the lead time of the of the single work center constructed for simulation.

Some operations are followed by an additional process step which does not require the capacity of the machine that performs the first process step. This can be modeled by an additional parameter providing the time of the second process step. An example of this kind of machine is an oven requiring additional chilling after the bake process. Assuming that the second process step is not subject to any capacity constraints a second infinite server can be constructed for the additional operations. Otherwise the second work center should be modeled as a work center of its own. These types of machines are quite similar to sequence work centers. Thus, their performance measures can be computed by analogy.

Cluster Tools are complex machines that allow to process several successive process steps without the interference of an operator. Typically, a

material handling system like a robot, for example, moves the parts from one step to the next. The tool needs its own control mechanism for scheduling the parts and the movements of the robot.

For the analysis via simulation complex cluster tools can be realized by sub-models that are to be generated automatically on the basis of parameters. The simulation expert defines which parameters are needed for a proper transformation of the real-world tool into the simulation model. The parameter suppliers have to enter the values for the parameters.

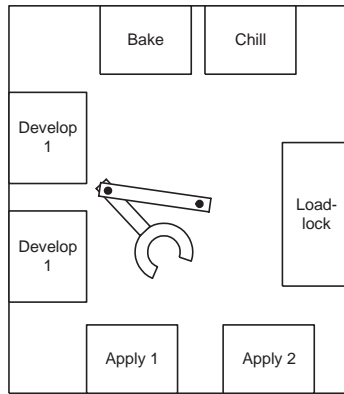


Figure 2.6.: Schematic view of the real-world photo cluster

As an example, figure 2.6 shows a typical cluster tool of a semiconductor line. The sample cluster tool is a photo cluster, i. e. its two tasks are the application of resist onto the wafers and the development of exposed wafers. The robot is shown in the middle of the cluster. Parts are started by placing them into the load-lock and registering them with the tool. Two disjunct process flows can be performed in this tool:

- *Apply resist.* A part is taken from the load-lock and is placed into one of the apply stations by the robot. There are two of them, but they might be dedicated to certain layers. After the resist has been applied, the part is put into the oven and then into the chilling station. Finally, it is put back into the load-lock for unloading.
- *Develop.* A part is taken from the load-lock and is placed into one of

the develop stations. After the develop process it is put back into the load-lock for unloading.

Parameters that are needed to model this kind of cluster tool are the number of apply, develop, bake, and chilling stations, their dedication to certain products and the processing time for all steps. The develop stations of the cluster can be modeled as shown in figure 2.7. The dispatcher routes the parts to either of the develop stations, depending on the technical requirements of different products. It should be modeled as an infinite server with zero processing time. Note, that the number of develop stations can be a parameter, as well, as another cluster might be equipped with three develop stations.

The apply stations can be modeled in analogy to the develop station (figure 2.8), i. e. a dispatcher of infinite capacity with zero processing time. If the apply station is the bottleneck, the bake and chill operations can be combined into one station in the simulation model. Note that this part of the cluster process is similar to the modeling of a sequence work center. As before, a tandem queue of a trigger station and a station for the remaining processing time are needed. If one of the bake or chilling stations is the bottleneck of the processing sequence, they have to be modeled by different stations. Whether queueing — either with finite or infinite buffer capacity — is permitted in this case depends on the special cluster and might ask for additional parameters of the model transformation.

The performance measures of such a cluster tool center can be computed as follows:

- *Utilization* (net/total). The utilization of the cluster work center is

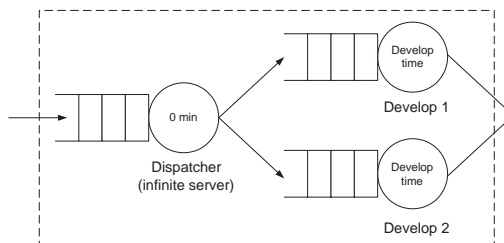


Figure 2.7.: Sub-model for cluster develop stations

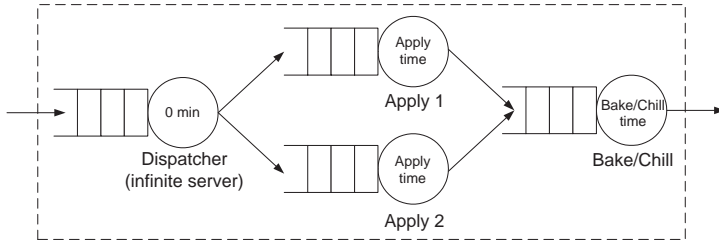


Figure 2.8.: Sub-model for cluster apply stations

the maximum utilization (net/total) of all of the work centers in the sub-model. As the dispatchers are modeled as infinite servers, their utilization equals zero.

- *Work-in-process.* The work-in-process of the complete cluster is the sum of all work-in-process inventories at all stations of the sub-model.
- *Lead time.* The lead time of the cluster work center is the weighted sum of the lead times at the work centers in the sub model. The weighting factors are the number of visits at the stations.

In modern semiconductor manufacturing environments the surface of wafers are structured with up to 20 or even 30 layers. Each layer normally needs an apply and a develop process step. As cluster tools are quite expensive and the processing times for the application of resist and for the development are rather short only few tools can be found on the shop floor. This results in a highly re-entrant flow, leading to complex scheduling problems at the photo clusters; for example, 2 operations per layer, and approx. 25 layers result in virtually 50 queues in each of which parts of all product types may wait for operation.

2.7.2. Parameterized Model Transformation

Often different questions in production planning require different simulation models. The model of *integrated simulation* requires the parameterized model generation. This means that certain scenarios can be generated automatically from the data warehouse. Examples for parameters in the model generation process are:

- *Different planning horizons.* Section 1.3.1 introduced different planning levels, each of which is associated with a typical period of planning, e.g. weekly, monthly, or yearly intervals. Simulation models for different levels in general require different models: With planning future capacities on the tactical level, it is not important – or even inadequate – to reflect tool dedication³ in the simulation model. But in short-term planning it is quite important to reflect dedicated machines in the simulation.
- *Load sizes.* Batch processing environments always face the problem of determining the optimal batch size. If transport batches are of random size and arrive randomly at a tool the question is when a batch run is to be started. If it is started early, batches arriving later have to wait and the tool's capacity might not be used completely. If a production batch is started late then the parts that arrive early at the tool have to wait long so that their lead time increases. Normally, there is a maximum, a minimum, and an average production batch size at a tool. Simulation models might also include threshold strategies, i.e. production at a tool is not started before at least n parts are ready for production. If the real-world model knows about these different load sizes and strategies, it can be transformed to different models.
- *Service type, dispatching.* Several parts arriving at the tool raise the question which one to process next. Several strategies, like first come first serve (FIFO), service in random order (SIRO), priorities, or routing probabilities can lead to a different performance of the production system.

If more than one tool is available then the decision which one to take for which batch has to be made as well.

- *Volume plan.* Different scenarios to be generated on the basis of the volume plan include the different volumes and product mixes as they can be found in the different periods of a volume plan. For example, a steady-state analysis could generate a model for a certain week of the volume plan. Other scenarios can be generated by different aggregations (for example weekly, monthly, yearly) of a given volume plan.

³Tools are specialized for certain products, i.e. not all tools of a work center are qualified for the same set of products.

- *Reliability.* The reliability states how long a tool is available for production on the average. Often, different sources for the reliability data exist, e.g. tool databases that monitor the state of the tools or plan data. Different sources of this data lead to different scenarios.
- *Size of a model.* The normal model generation process generates a model for a complete production line. But sometimes such a model gets too large for the simulation. In this case part of the model could be generated. For example, estimating the performance measures of a production line based on a queueing network works well. The same model in a discrete event simulator might take too long. But a sub-model containing just some work centers might help to evaluate different dispatching strategies which in turn are not tractable for queueing networks.
- *Yield.* Especially in large semiconductor facilities rework and yield play a very important role for the performance measures. Generating a model based on historic data might vary much from a model generated on the basis plan values and assumptions for the future.
- *Number of tools at a work center.* The number of tools that varies with time has the most striking effect on capacity. New products and changing demands may ask for additional tools or might make tools superfluous if production volumes decrease.

In addition to these scenarios derived from daily planning tasks some others can be generated. Basically, every parameter that can be derived from different sources (planning assumptions or monitoring systems) leads to a new scenario. This includes the variance of down and/or processing times, for example.

2.7.3. Processing Incorrect and Incomplete Data

The main problem during automatic model generation is incomplete and/or inconsistent information caused by incomplete or faulty data entered by employees or taken from shop-floor-control systems, respectively. During the parameter collection these problems are tried to be avoided by means of integrity checks and data cleansing (see sections 9 and 5.6). Nevertheless, *integrated simulation* needs a concept to cope with missing or incorrect data.

This step is part of the model transformation process because only if the simulator and the scenario are known, the model transformer knows which parameters are missing since not every simulator uses all information of the real-world model. For example, complex dispatching rules might be specified for a discrete event simulator, but are not needed for queueing models assuming a FIFO service order.

The intention is to create a model that can be simulated under any circumstances so that the model build cycle can be started as early as possible. The system does not have to wait until all data input has been completed. Thus validation and the steps of model creation can run in parallel and time is saved. The same idea is presented in [Try94].

In many cases the missing data is not substantial for the outcome of the simulation. If just a small percentage of rather insignificant parameters is missing, the outcome of the simulation using reasonable default values and assumptions can still be valuable. Even if the simulation run produces wrong results the data of the results and the assumptions can effectively be used in the debugging and validation process:

- The logged default values for missing parameters show directly where additional input is required.
- Average overall performance shows the magnitude of deviation of the simulation model from the real world. Thus the user may distinguish between systematic errors or parameters that are slightly wrong.
- Bottleneck charts quickly reveal tools with wrong parameters, as normally rough ideas exist of how much the tools are utilized.

Several techniques can help to generate simulation models on the basis of faulty or inconsistent data, for example:

- *Default values.* Missing parameters can be set to default values that affect the outcome as little as possible.
- *Heuristics.* Having domain knowledge heuristic checks can warn if parameters are likely to be out of range, for example if an employee swaps MTTR and MTBF values, the reliability of the corresponding tool is likely to be close to 0. This can be spotted automatically.

All problems and assumptions occurring in the model generation process need to be logged as the user has to be aware of the fact that the results of

the simulation might not be correct. This log must be presented to the user together with the simulation results and parameters so that he can make a personal judgement of the severity of the inconsistent and faulty data.

A more sophisticated system could scan the simulation model for suspicious parameters or obvious errors and start a sensitivity analysis on the missing parameters. Thus the user can get an idea of the severity of the problems found during model generation.

2.7.4. The Simulator

As complex production lines are heavily influenced by random effects there are mainly two different methods to take these into consideration, discrete event simulation or analytical performance analysis as shown in section 1.4. The model of *integrated simulation* requires fast response times as the production planner uses the simulation interactively in the planning process. Response times of several minutes are not acceptable, especially if simulation results are to be used for on-line control of the production process. Thus, only analytical performance analysis is suitable [Zis99]. The author describes the mathematical background of a queueing network analyzer for $G^X/G(b, b)/c$ systems.

Figure 2.9 shows how the simulator is integrated into the simulation process. There are three different levels of abstraction, on the lowest the simulator performs the simulation.

Including the step from the simulator API to the underlying mathematical representation, i.e. to the vectors and matrices used in the queueing theory formulae, the complete hierarchy of model generation and transformation is shown in figure 2.9. The mathematical representation allows to solve the systems of equations efficiently that are needed to calculate the performance measures of the simulated production line.

To summarize, the steps down the modeling hierarchy are the generation of a model for the real world, the transformation of that model to a model described in the API (application programming interface) of the simulator and finally the mapping to the internal data structures of the simulator, i.e. in the case of *integrated simulation* to queueing theory formulae. Object-oriented methods in all of these steps allow the efficient mapping from the real-world production facilities to the queueing theory formulae.

On the way back up to the real world, the mathematical results have to be assigned to the appropriate objects so that they can be interpreted in

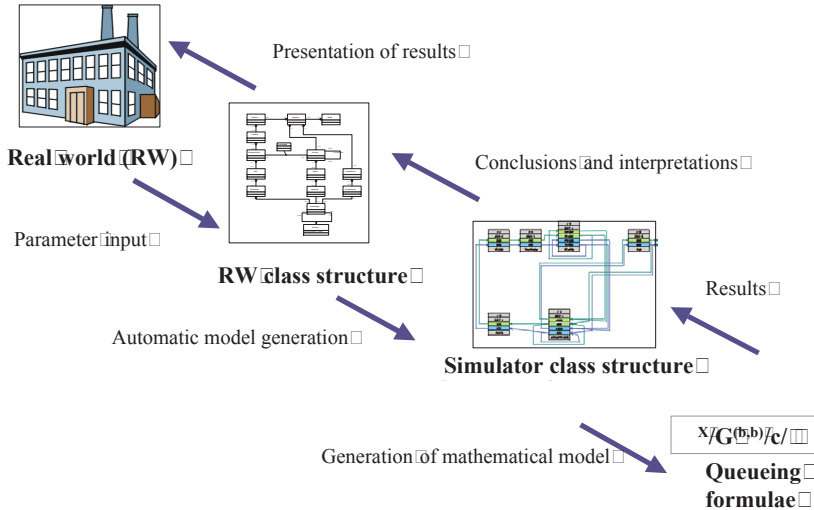


Figure 2.9.: Model transformation steps of integrated simulation

different contexts, i. e. results for auxiliary work centers have to be aggregated to performance measures of their original work centers. Finally the engineers, planners and managers are presented with the simulation results customized to their specific needs and business processes.

2.8. Model Validation

It is obvious that useful, reliable planning has to be based on valid models. Validity means that the model correctly represents reality with respect to the features considered, i. e. the behavior of reality concerning those aspects that are to be examined by a simulation study is observed in the same way in the model (see section 1.4 for details). Many authors have emphasized the need for validation like Law [LK91] and Robinson et al. [RGWAC99, CRW99]. As the models within *integrated simulation* are the basis for an ongoing planning procedure, their validity has to be maintained over time. Permanent changes with respect to tool dedication, volume plans, process flows, yield assumptions etc. require an ongoing validation of the simulation

models. In [CRW99] some steps to get to a valid simulation model are proposed:

- *Presentation of mid-way results.* The earlier possible misunderstandings and errors are detected the better.
- *Documentation of all assumptions.* Often simulation projects are executed over a long period with changing developers. Modeling assumptions have to be available all the time such that results can be interpreted correctly.
- *Sensitivity analysis.* Parameters effecting the results most should be identified and checked first, especially if they are taken automatically from other systems.
- *Validation against factory data.* The comparison of actual data can reveal discrepancies between the model and reality.
- *Walk-through.* Explaining assumptions and how the model works to engineers can reveal misunderstandings occurring in early modeling phases.

The most important point is the validation against factory data as for large models this is the only efficient method. This kind of model validation can take place on two different levels:

- Model parameter validation
 - ▷ manually provided parameters
 - ▷ automatically integrated parameters
- Performance measure validation

In the end the performance measures have to be valid, but especially for large simulation models it is quite difficult to derive the causes for calculated performance values diverting from reality. Thus the idea is to start earlier on the level of input parameters. Only if these are correct, the results can also be correct. In general, parameters cannot be directly compared to corresponding actual values. The latter have to be estimated by statistical methods like time series, regression analysis, and the estimation of means and variations.

Parameters entered manually can be compared with statistical estimates taken from shop-floor-control systems. The reason why the statistically derived estimates are not automatically used for simulation is that the simulation should be based on planning assumptions. Taking real-world estimates leads to the problem of small samples biasing the planning parameters. Moreover, the observations to estimate parameters often need to be calculated from shop-floor-control data, see section 11.2. But a comparison provides an overview on how far the planning parameters divert from reality.

For example, to check whether the cycle times entered by engineers are up-to date, means can be computed from claim times tracked by the shop-floor-control system. Comparing the values taken from reality to their theoretical plan parameters can show deviations. Moreover, down times can be derived from or compared to data from systems tracking tool state. Thus the reliability of tools can be checked.

Parameters that are automatically imported into the simulation model lack this kind of consistency check. But especially in this case planners have to be careful as pointed out in [RGWAC99]. Data taken from external systems might include systematic and statistical errors. This is especially the case if the data extracted for parameter input or validation has never been used before for planning purposes, which is often the case for logistical data. Thus the only chance is to apply heuristic range checking methods to the imported parameters, for example a chart showing the yield drop along the process flow. Planners can judge intuitively if the parameters presented that way are sensible assumptions for simulation.

To validate performance measures like throughput, lead time, work-in-process, etc. logistical control charts can be used. Actual performance measures have to be compared with simulation results. Further details on this point are discussed in section 11.4.

There are some problems in comparing parameters and performance measures to factory data. Unfortunately, this procedure is only possible for near to mid-term planning as future parameters cannot be estimated from current production, i.e. it is always only possible to validate against current performance.

Yield assumptions can be derived from the statistical analysis of factory data, for example. The problem is to choose the right period for yield estimates: If the period is chosen too long then the assumptions might be outdated, if it is too short, then there might not be enough samples to derive statistically significant yield assumptions. Note that especially yield

assumptions are often based on a learning curve.

All the ideas presented above are reflected within *integrated simulation*. The approach to build simulation models even on inconsistent and incomplete data leads to an early understanding of problems and provides helpful insights. The documentation of the basic modeling assumptions (cluster tools, pipelining tools, etc.) and of the mathematical model can be found in this thesis. Moreover, each simulation run comes along with an event history listing all errors, inconsistencies, and default values set during the model generation.

Moreover, *integrated simulation* requires certain reports to check the validity of the model such that errors can be detected at a glance, for example. This includes reports showing discrepancies between the process flows imported from the shop-floor-control system and the manual input of process plans by the engineers.

Another means of validation are *back-on-the-envelope* calculations for estimating capacity. For example, with the tool parameters and the process plan given, rough estimates on the magnitude of the throughput rates can be computed. These are to be placed next to the input parameters such that every engineer can directly see the influence of parameter modifications.

2.9. Business Process Capacity, Lead Time, and Work-in-Process Planning

The model requires a tight integration of the simulation (results) into the planning and decision processes of the company. This means that every user should be provided with the topical simulation results he needs at the correct aggregation level. The essential purpose of the business process that comes along with *integrated simulation* is to determine a feasible production schedule with respect to capacity, lead time, and work-in-process forecasts. A new volume plan is feasible if the demands for each period can be satisfied with the capacity available in that period. The employees involved are capacity planners whose concerns are satisfied by bottleneck charts of future periods. Manufacturing and finance managers will focus on the development of lead time and work-in-process. Capital planners get hints of where to invest, i. e. which tools to buy, in order to meet capacity demands and/or lead time constraints. The outcome is a highly condensed report that specifies which actions are to be taken if the plan is infeasible. This is the so-called

commented standard report.

As an example, the swim lane diagram in figure 2.10 shows the process flow of this business process as it might be applied in a company. The four protagonists of the process are:

- *PMC*. The Production management committee supplies the demand in form of a first version of the production schedule. It is mainly based on customer demands.
- *Wafer logistics*. These are the manufacturing representatives that take part in the process.
- *Capacity planning*. The capacity planning department is responsible for a feasible production schedule.
- *Simulation team*. The simulation team provides the infrastructure for simulation, i.e. the system that supplies the data and performs the analysis of the production line's performance measures.

The following is a step by step explanation of the swim lane diagram:

1. The PMC releases a new demand plan.
2. When the manufacturing department receives this plan, two actions are triggered that can be run in parallel:
 - a) The new demand plan has to be read into the data warehouse.
 - b) The state of the simulation model has to be checked. Normally, the model should be validated all the time. Certain indicators allow to judge on the quality of the model, for example the age of parameters, the number of approved parameters, consistency checks of model validity, etc. This step is supposed to avoid to start with an outdated model which just would be a waste of time.
3. If the prerequisites for the simulation have been finished the simulation is started. Different scenarios are evaluated for each period of the demand plan.
4. The result reports are generated and the commented standard report is created (see section 8.5.2).

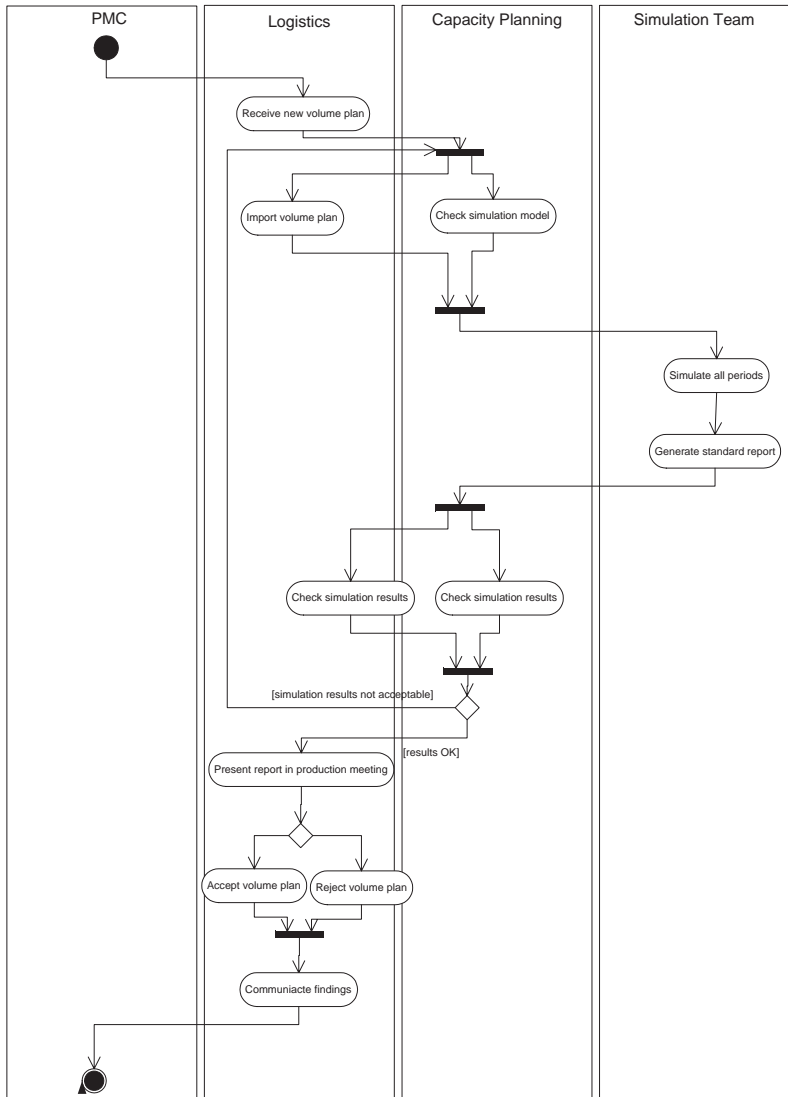


Figure 2.10.: Process of capacity, lead time, and work-in-process planning

5. The manufacturing and the capacity planning department check the outcomes of the simulation.
6. If both departments agree that the results are not acceptable the model has to be updated and/or the demand has to be adjusted.
7. If the results are accepted, they are presented to a wider audience in a production meeting.
8. This audience finally decides on the acceptance of the plan. The findings are reported to the PMC.

Apart from standard reports for the simulation results a more condensed report is needed for management meetings. Among the huge number of evaluated performance measures results have to be selected and commented. This yields a commented report which should be standardized due to usability purposes. Both the assumptions and the results of the simulation have to be part of this report. This report focuses directly on the critical points. Moreover, the actions to be taken are to be added to the report. Thus the current state of actions can be discussed in the following meeting.

2.10. Summary

Integrated simulation offers a lot of opportunities. It defines an environment and processes that allow to provide high quality simulation studies for production planning purposes. The model is designed to support an ongoing planning process in contrast to one-time simulation projects. This is especially important for production environments which are subject to frequent changes as they can be found in semiconductor industry. Figure 2.2 shows the production types (see section 1.3.2) and planning levels (see section 1.3.1) for which *integrated simulation* is proposed. The granularity of the model is suited best for tactical planning. Operational planning has to consider set-up states, scheduling policies, etc. These cannot be handled by the queueing network analysis. For strategic planning a queueing model is probably too detailed. Normally, for future products only the essential process steps are known, and there are only rough estimates for scrap and rework rates. However, it is possible to model strategic scenarios based on overall assumptions. Special modeling algorithms could construct feasible queueing networks from simple estimates. With an overall yield given, routings could be constructed

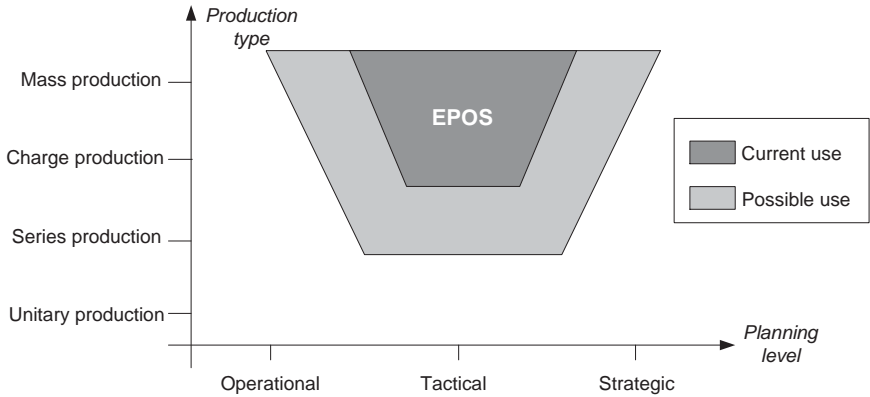


Figure 2.11.: Use of *integrated simulation*

that distribute the yield loss linearly along the production line. This justifies the top right corner of the *possible use* polygon. The top left corner is justified by plan/actual comparisons and logistical quality charts that allow to see whether the logistical parameters of the line are within their natural bounds.

Queueing network analysis relies on a statistical steady-state analysis. Thus it is not applicable to unitary production, where only a single product is produced. The more parts are run through the manufacturing line the better the model fulfills its purpose.

Integrated simulation is supposed to improve the quality of production planning. With the help of sophisticated data management and calculation procedures the planner is able to concentrate on his actual task — thinking about the results to find feasible and optimal production schedules.

It is shown that data acquisition and management play an important role in simulation projects. Thus *integrated simulation* provides special processes for manual and automatic data acquisition. All planning data is stored in a central data warehouse, which offers the following advantages:

- Complete and consistent set of planning parameters
- Easy report generation for documentation purposes

- ▷ Transparency
- ▷ Preparation/execution of audits
- Comparison of planning parameters for different production lines (identity)
- Basis for all kinds of calculations (simulation, optimization, spreadsheet analysis, *back-of-the-envelope* algorithms, and other planning methods like global ATP, etc.)
- Well-defined processes and responsibilities for parameter input
- Avoidance of redundancy

Concerning data acquisition special emphasis has been put on parameters that have to be entered manually as this is the most obvious source of errors in the planning process. The properties of the approach described in this section are:

- Manual parameter input is a well-defined process.
- Distributed responsibilities: Data is gathered at the source of information.
- Easy incremental updates of machine parameters
- On-line data input and access (24-hour availability)
- Easy and shared access to machine parameters (world wide)
- Automatic plausibility checks
- Approval process supported by electronic signatures
- Easy software roll-out

Integrated simulation requires the automatic generation of models for different planning scenarios and has to apply standard modeling techniques. Additionally, fast simulation methods that are completely integrated into the environment are necessary to yield the following characteristics:

- *Automatic generation.* It allows non-experts to make use of elaborated simulation methods.

- *Fast model generation.* Models based on current parameters are always available.
- *Fast evaluation.* The speed of evaluation makes it possible to calculate many scenarios and is essential for interactive planning
- *Integrated simulator.* The integrated simulator uses a secure communication via API's applying well-defined protocols. All error messages stored in data warehouse and thus help to understand the results. Advanced mathematical methods and integrated optimization algorithms that cannot be realized by spreadsheets become available.

Simulation results have to be accessed by planners, managers, etc. Therefore results are stored in the central data warehouse in combination with the parameters. This offers the following advantages:

- Automatic report generation as dynamic and/or static web sites
- Server based distribution
- Easy aggregations and analysis by SQL or data mining tools
- Logistical process control (LPC)

The opportunities *integrated simulation* offers by integrating e-business technologies and advanced mathematical methods can only be fully exploited, if all planning processes of a company are integrated into the approach. All planners that rely on the data stored in the central data warehouse have to use it. Otherwise redundancy and inconsistency cannot be avoided and efficiency is not improved. Thus *integrated simulation* requires a re-engineering of planning processes.

Chapter 3

Queueing Network Analysis

Queues develop in situations when in front of a service station more units per time interval arrive than can be processed during that interval. In many cases such systems do not only exist isolated, but in networks of more than one queueing system. The analysis of these networks in terms of performance measures can be classified as follows:

1. Exact analysis
2. Approximation methods
3. Simulation and related techniques

Exact results for queueing networks exist for Markovian systems. For a restricted class of so-called Jackson networks [Jac57] results for the equilibrium distribution of the number of units exist. The lack of success in obtaining exact solutions for general networks has motivated the development of approximations for performance values. The third technique, mainly discrete event simulation, yields the possibility of modeling queueing networks in great detail. The main drawback of this method are the computational requirements needed to derive statistically relevant information, which often limits the number of alternatives that can be evaluated.

This chapter introduces a network approximation method to analyze a broad range of queueing networks. Formally, performance measures for $G^X/G(b, b)/c$ systems (bulk arrivals and batch processing) can be evaluated.

3.1. Introduction

The foundations of queueing theory were laid in 1908 when A. K. Erlang, a Danish engineer, published the *The Theory of Probabilities and Telephone Conversations*. Erlang observed that the telephone system could be characterized by a Poisson input process, exponential holding time, and multiple service channels. Using the standard classification of queueing systems, introduced 1951 by D. G. Kendall, this system is denoted by $M/M/c$. In 1969 Bhat published the well-known article *Sixty Years of Queueing Theory* [Bha69] in which he comments on the achievements and shortcomings of queueing theory. Throughout these sixty years the motivation has shifted from "practical congestion problems" to "more general models which could be used in more complex situations".

In 1957 Jackson [Jac57] presents one of the first results of steady state distributions for queueing networks. The nodes in the so-called Jackson network are $M/M/c$ stations which can be treated as if they were isolated stations. This approach is not exact for more general types of networks. In 1973 Kobayashi and Reiser [RK73] present the parametric decomposition methodology (see section 3.12) which yields approximations for performance measures in general networks. Permitting bulk arrivals and batch processing further complicates modeling the flow of materials through the network. Nevertheless, approximations for quite general types of queueing networks which can successfully be deployed in production planning have been derived by now.

A thorough introduction to queueing theory and network decomposition methods is beyond the scope of this work. This section just introduces the notation and basic definitions which are used throughout this chapter. Readers who are familiar with stochastic processes and the theory of queues should directly forward to section 3.2.

Stochastic processes, the foundation of queueing theory, are discussed in the monographs by Karlin [KT75, KT81] or Chung [Chu67], Cox and Miller [CM70], or Cinlar [Cin75], for example. For detailed information on queueing systems the reader should refer to Kleinrock [Kle75, Kle76, KG96], Gross and Harris [GH74], Lavenberg [Lav83a], or the article by Saaty [Saa57]. Bhat [Bha69] gives an overview over different areas of research.

In the following the basic definitions needed throughout this chapter are stated (the reader familiar with these concepts might directly forward to section 3.2).

Definition 3.1.1 (Expectation and variance) *Let X be a random variable and $f(x)$ its continuous density. If the expectation $E[X]$ of X exists (see [MP90]) it is given by*

$$E[X] = \int_{\mathbb{R}} xf(x)dx. \quad (3.1)$$

If the variance of X exists (see [MP90]), it is defined by

$$\text{Var}[X] = E[(X - E[X])^2]. \quad (3.2)$$

The coefficient of variation which will be defined in the following is a means to compare the amount of variability of different distributions of random variables.

Definition 3.1.2 (Coefficient of variation) *The coefficient of variation (CV) of a random variable X is defined to be the quotient $\sqrt{\text{Var}[X]}/E[X]$ of the standard deviation and the expectation of X for $E[X] \neq 0$. The squared coefficient of variation (SCV) is defined by*

$$C^2 = \frac{\text{Var}[X]}{E[X]^2}. \quad (3.3)$$

Definition 3.1.3 (Stochastic process) *Let $(X_t)_{t \in T}$ be a family of random variables with values in E where t is a parameter running over a suitable index set T . $(X_t)_{t \in T}$ is called a stochastic process with index set T and state space E . If E is denumerable the process is called discrete state process, otherwise continuous state process.*

Classifications of stochastic processes usually distinguish between different types of state spaces, index sets, and dependence relations between the random variables X_t . Fundamental stochastic processes with discrete index set are the renewal processes.

Definition 3.1.4 (Renewal process) *A sequence $(X_n)_{n \in \mathbb{N}}$ of independent and identically distributed positive random variables is called renewal process. It has an associated process of partial sums $(S_n)_{n \in \mathbb{N}}$ with $S_0 = 0$ defined by $S_n = \sum_{k=0}^n X_k$ ($n \geq 1$). X_n represents the lifetime of some unit, S_n the time of the n th renewal. A renewal counting process N_t , $t \in \mathbb{R}^+$ counts the renewals in the interval $[0, t]$, formally $N_t = n$ for $S_n \leq t < S_{n+1}$, $n \in \mathbb{N}$.*

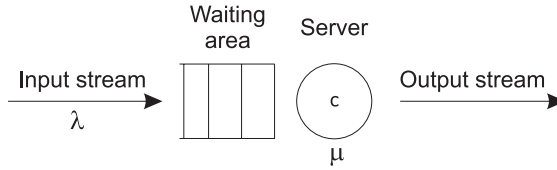


Figure 3.1.: General notation of a queueing station

Figure 3.1 shows the graphical representation of a general queueing station. Since the notation of input and service processes has been well established (see [Mac61]), the common symbols are used throughout this chapter, as well. The inter-arrival times of units entering the system are a renewal process $(I_n)_{n \in \mathbb{N}}$ with arrival rate $\lambda = \frac{1}{\mathbb{E}[I]}$. The service rate is denoted by μ . With c as the number of available servers the utilization of the station is given by $\rho = \frac{\lambda}{\mu c}$. A system containing a positive amount of variability can reach a state of equilibrium only if ρ is smaller than one¹. This result is quite obvious and is assumed unless noted otherwise.

A fundamental result of queueing theory reveals the relation between the average number of units in a system and the mean time the units spent in the system:

Theorem 3.1.1 (Little's law) *Let N_t be the process of numbers of units in a queueing system at time t and Q_n the time of the n th unit in the system. λ should denote the input rate. The following equation holds*

$$\mathbb{E}[N] = \lambda \mathbb{E}[Q] . \quad (3.4)$$

This theorem can be proved under quite mild assumptions (see [Lit61]).

3.2. The Model

Throughout the following sections a queueing network model is presented. The basis of this model is a system called AMS (Analytical Modeling System) which was developed by Hanschke [HZ97b] and Zisgen [Zis99] (see also [Oph96] for implementation details). To approximate performance measures

¹A stationary state with $\rho = 1$ can only be reached for $D/D/c$ systems

AMS relies on the parametric network decomposition approach (see section 3.12). The nodes of the network called work centers are $G^X/G(b,b)/c$ queueing stations, i. e. they allow for batch processing and bulk arrivals. An implementation and an object-oriented description of this model is presented in chapter 6.1. The following sections focus on the mathematical background of the model. Intermediate results will be developed step by step, such that finally the overall performance measures like work-in-process or the cycle time for the complete model can be calculated.

3.3. Input Parameters

The model consists of three sets for the product types, operations, and work centers, \mathcal{U} , \mathcal{A} and \mathcal{W} , respectively. A work center $w_k \in \mathcal{W}, k = 1, \dots, W$, can be characterized as a tuple $w_k = (c_k, r_k, MTT R_k, C^2[D_k], b_k)$ with

- number of tools c_k ,
- reliability r_k ,
- mean down time $MTT R_k = E[D_k]$,
- squared coefficient of variation $C^2[D_k]$ and
- batch size b_k of all process steps being performed at w_k .

Each work center is associated with a set of process steps to be performed at the work center. Generally, a process step $a_m \in \mathcal{A}, m = 1, \dots, A$, is a tuple $a_m = (E[S_m], C^2[S_m])$ with

- mean process time $E[S_m]$ and
- squared coefficient of variation of the process time $C^2[S_m]$

as input parameters. The model requires all batch sizes of a work center to be equal as the calculation of a work center's input batch size is done on work center level².

Process steps always belong to exactly one product type, i. e. there is a $1 : n$ relationship between process steps and product types. In other words, a product type consists of a set of process steps which are needed to produce

²A later model not yet published does not contain this constraint any longer.

a good part of that specific type. A product type u_l for $l = 1, \dots, U$, is a tuple $u_l = (E[I_l], C^2[I_l], b_l, C^2[B_l])$ which is completely specified by the following input parameters:

- the expected inter-arrival times $E[I_l]$ of parts of product type u_l at the source operation of this product type,
- the squared coefficient of variation of the inter-arrival time $C^2[I_l]$,
- the mean batch size $b_l = E[B_l]$ of parts entering the system, and
- the squared coefficient of variation of this batch size $C^2[B_l]$.

The order in which the process steps are to be performed is specified by a directed, weighted graph $G = (\mathcal{A}, \mathcal{R})$ with the set of process steps \mathcal{A} and a set of routes \mathcal{R} serving as edges in the graph. For each route the probability that a part moves from a preceding process step a_m to the successor a_n must be specified by $p_{m,n}$. In the resulting routing matrix $P = (p_{m,n})_{m,n=1,\dots,A}$, only positive probabilities between process steps that belong to the same product type are allowed. Moreover, for each product type u_l exactly one source process step $a_{\text{source}(l)}$ with $\sum_{n=1}^A p_{n,\text{source}(l)} = 0$ and one sink process step $a_{\text{sink}(l)}$ with $\sum_{n=1}^A p_{\text{sink}(l),n} = 0$ is allowed. Sink process steps are the only process steps which are not associated with a work center. All other process steps need to be assigned to exactly one work center. To enable modeling of scrap the sum of all outgoing routes at a process step can be less than 1. Thus, the scrap at process step a_m is defined by

$$\text{scrap}_m = 1 - \sum_{n=1}^A p_{m,n} \quad \text{for all } m \in \mathcal{A}. \quad (3.5)$$

The structure of the queueing model is expressed by the functions shown in table 3.1 that translate indices of any type into indices of another type.

Comment: In this work the two terms operation and process step are used for different classes of objects (see part II for details). An operation is the abstraction for some piece of value-add work that has to be performed in order to create a finished good. An operation is independent of a product type and is not specified to be carried out on a specific work center. In contrast to this process steps are defined for a specific work center and product type.

$w(m)$: $\{1, \dots, A\} \rightarrow \{1, \dots, W\}$	returns k , if process step a_m is performed on work center w_k ,
$p(m)$: $\{1, \dots, A\} \rightarrow \{1, \dots, U\}$	returns l , if process step a_m is of product type u_l .
$sink(l)$: $\{1, \dots, U\} \rightarrow \{1, \dots, A\}$	returns m , if process step a_m is the sink in the process flow of product type u_l .
$source(l)$: $\{1, \dots, U\} \rightarrow \{1, \dots, A\}$	returns m , if process step a_m is the source in the process flow of product type u_l .
$pred(r)$: $\{1, \dots, R\} \rightarrow \{1, \dots, A\}$	returns m , if routing r connects process step a_m to process step a_n .
$succ(r)$: $\{1, \dots, R\} \rightarrow \{1, \dots, A\}$	returns n , if routing r connects process step a_m to process step a_n .

Table 3.1.: Structure of the model: Mappings of indices

Class	Set	Object	Cardinality	Indices
Work center	\mathcal{W}	w	$ \mathcal{W} = A$	k, (i, j)
Process step	\mathcal{A}	a	$ \mathcal{A} = A$	m, n
Product type	\mathcal{U}	u	$ \mathcal{U} = U$	l, (i, j)

Table 3.2.: Naming convention of model artifacts used in this section

Table 3.2 shows the naming convention for the classes of the model, the corresponding objects, containers (sets), their cardinalities, and the indices used whenever possible.

3.4. Demand and Bill-of-Materials

To model multi-stage production processes, relationships between products — these might be either partial products or end products — are stored in the bill-of-materials (BOM). It is specified by a (Gozinto) graph (see

[GT00], [Sch99], [Sch93], [GT00], [Ada97], for example) which shows how many parts of product type u_i are needed to produce one part of product type u_j . This graph is stored by means of the matrix of direct demand coefficients $D = (d_{i,j})_{i,j=1,\dots,U}$. The primary demand specifies the demand of products which are to be sold and not to be used as intermediate products in the manufacturing process. It is given by the vector $d = (d_1, \dots, d_U)$. The demand vector d and the matrix D allow to calculate the secondary demand $x = (x_1, \dots, x_U)$, a vector specifying how many parts of each product type are needed to fulfill the production of the primary demand. The secondary demand can be calculated by solving the following system of linear equations (in matrix form):

$$x = (I - D)^{-1}d \quad (3.6)$$

where I denotes the identity matrix. Solving this system allows to handle converging, diverging, and general product structures.

3.5. Number of Visits

In case that process step a_m is not a source its number of visits is defined by $e_m = \sum_{n=1}^A p_{n,m} e_n$. Otherwise the number of visits is defined to be 1. As the routing graph is a general graph modeling scrap and rework, the number of visits at each process step has to be calculated by solving the system of linear equations

$$e_m = e_{\text{source},m} + \sum_{n=1}^A p_{n,m} e_n \quad \text{for } m = 1, \dots, A \quad (3.7)$$

with

$$e_{\text{source},m} = \begin{cases} 1 & \text{if } m = \text{source}(p(m)) \\ 0 & \text{otherwise} \end{cases} \quad (3.8)$$

To formulate this system of equations in matrix form it has to be rearranged to

$$\begin{aligned} e_m - \sum_{n=1}^A p_{n,m} e_n &= e_{0,m} \quad \text{for } m = 1, \dots, A \\ \Leftrightarrow \sum_{n=1}^A (\delta_{m,n} e_m - p_{n,m} e_m) &= e_{0,m} \quad \text{for } m = 1, \dots, A \end{aligned} \quad (3.9)$$

where

$$\delta_{m,n} = \begin{cases} 1 & \text{if } m = n \\ 0 & \text{if } m \neq n \end{cases} \quad (3.10)$$

denotes the Kronecker symbol. This yields $(I - P^T)\vec{e} = \vec{e}_{\text{source}}$ as a system to determine the number of visits.

The throughput yield γ_l for product type u_l is defined as the number of visits at the sink $a_{\text{sink}(l)}$ of this product type:

$$\gamma_l = e_{\text{sink}(l)}. \quad (3.11)$$

Using (3.6) the yielded secondary demand is determined by

$$x_l^\gamma = \frac{x_l}{\gamma_l} \quad \text{for } l = 1, \dots, U. \quad (3.12)$$

3.6. Product Mix

The total demand λ , specified in parts per time unit, is the sum of each product's secondary demand

$$\lambda = \sum_{l \in \mathcal{U}} x_l^\gamma. \quad (3.13)$$

This is the input rate of the whole model.

To specify the input rate for each product type the total input rate has to be weighted by the product mix α_l which is defined as

$$\alpha_l = \frac{x_l^\gamma}{\lambda} \quad \text{for } l = 1, \dots, U. \quad (3.14)$$

3.7. Probability of Good Parts

In this step the probability g_m that a part which starts at process step a_m finally reaches the sink and becomes a good part is calculated. This result will later (see section 3.17) be needed in order to calculate the lead time of good parts from that process step a_m along the process flow to the sink of the network. The probability g_m can be defined recursively which leads to a system of linear equations which is quite similar to the one noted in (3.7):

$$g_m = g_{\text{sink},m} + \sum_{n=1}^A p_{m,n} g_n \quad \text{for } m = 1, \dots, A \quad (3.15)$$

with

$$g_{\text{sink},m} = \begin{cases} 1 & \text{if } m = \text{sink}(p(m)) \\ 0 & \text{otherwise} \end{cases}. \quad (3.16)$$

In matrix form this system can be written as $(I - P)\vec{g} = \vec{g}_{\text{sink}}$. This reveals the similarities and differences between the g_m and e_m (see section 3.5). Obviously the right hand side of the system is either the vector of sources (in case of the number of visits) or the vector of sinks (in this case). Moreover, to calculate the number of visits the transposed matrix P^T is used as the weighted sum of predecessors is to be computed. The throughput yield γ_l of product type u_l can be expressed by either the number of visits (see equation (3.11)) or by $g_{\text{source}(l)}$ which is the probability that a part which starts at the sink of the network reaches the sink as good part.

3.8. Number of Visits at Work Centers

To calculate the number of visits at each work center the set $\mathcal{A}_{l,k}$ denoting the set of all process steps of product type l which are performed at work center k is introduced. With that the relative number of visits of product type l at work center k is given by

$$\hat{e}_{l,k} = \alpha_l \sum_{m \in \mathcal{A}_{l,k}} e_m \quad \text{for } l = 1, \dots, U \text{ and } k = 1, \dots, W. \quad (3.17)$$

The number of visits at work center k for all product types can then be calculated by

$$\hat{e}_k = \sum_{l \in \mathcal{U}} \hat{e}_{l,k} \quad \text{for } k = 1, \dots, W. \quad (3.18)$$

Given the number of visits at each work center the transition probabilities between work centers can be calculated. In order to do this certain subsets of process steps in \mathcal{A} have to be defined: Let \mathcal{A}_u be the set of process steps of product type u . Similarly, \mathcal{A}_{u,w_i} denotes the set of process steps of product type u at work center w_i . The probability $\hat{p}_{i,j}$ to get from work center w_i to work center w_j is given by

$$\hat{p}_{i,j} = \frac{\sum_{u \in \mathcal{U}} \sum_{r \in \mathcal{A}_{u,w_i}} \sum_{s \in \mathcal{A}_{u,w_j}} \alpha_u e_r p_{r,s}}{\hat{e}_i}. \quad (3.19)$$

3.9. Service Times at Work Centers

It is normal to have different distributions of process times of process steps even at the same work center. This is reflected in the model by the definition of process steps: The process or service time of process step a_m is approximated by the first two moments of the random variable S_m . To be able to use the process time at work center level different service times have to be aggregated. The expected service time $E[\hat{S}_k]$ at work center w_k is calculated using the weighted service time of the assigned process steps in relation to the work center's number of visits

$$E[\hat{S}_k] = \frac{\sum_{u \in \mathcal{U}} \sum_{a \in \mathcal{A}_{u, w_k}} \alpha_u e_a E[S_a]}{\hat{e}_k} . \quad (3.20)$$

The squared coefficient of variation $C^2[\hat{S}_k]$ of the work center's process time is given by

$$C^2[\hat{S}_k] = \frac{\sum_{u \in \mathcal{U}} \sum_{a \in \mathcal{A}_{u, w_k}} \alpha_u e_a E[S_a]^2 (C^2[S_a] + 1)}{\hat{e}_k E[\hat{S}_k]^2} . \quad (3.21)$$

3.10. Completion Times at Work Centers

The expected completion time $E[\hat{C}_k]$ at work center w_k is the average time a batch needs to be processed at the work center including the time a batch possibly has to wait for the machine to be repaired in case of breakdowns.

Comment: Many values calculated are used at different aggregation levels throughout this section. The number of visits, for example, is calculated for process steps and then aggregated for work centers. To differentiate between basic and aggregated values the latter are marked with a symbol above the letter, e.g. the number of visits of process step a_m is given by e_m , the number of visits at a work center w_k is denoted by \hat{e}_k . Aggregations for work centers are denoted by a hat ($\hat{\cdot}$), process steps by a check ($\check{\cdot}$) and product types by a tilde ($\tilde{\cdot}$). The remaining lead time is denoted by a breve ($\breve{\cdot}$). For better readability this is only done when values appear on multiple aggregation levels. The input batch size X_k (see section 3.11), for example, only applies for work centers and is consequently denoted without any symbol.

Generally, the process of machine failures can be described by an alternating renewal process [Mül91] consisting of $(U_n)_{n \in \mathbb{N}}$ and $(D_n)_{n \in \mathbb{N}}$ which describe the intervals in which the work center is up and running and down because of machine failures, respectively. The reliability r_k of work center w_k is defined as

$$r_k = \frac{MTBF_k}{MTTR_k + MTBF_k} \quad (3.22)$$

where $MTBF_k$ and $MTTR_k$ refer to the expected values of the stochastic processes mentioned above, namely

$$\begin{aligned} MTTR &= E[D_1] \\ MTBF &= E[U_1], \end{aligned}$$

assuming that $(U_n)_{n \in \mathbb{N}}$ and $(D_n)_{n \in \mathbb{N}}$ are independent and identically distributed (iid). Let M_k be the number of failures during the processing of a batch at work center k . The completion time \hat{C}_k at work center w_k is defined as the sum of the service time and possible down times during processing

$$\hat{C}_k = \hat{S}_k + \sum_{\nu=1}^M D_{k,\nu} \quad (3.23)$$

with $D_{k,\nu}$ as the ν th down time. In this model the time between two failures is assumed to be exponentially distributed with parameter $\kappa_k = 1/MTBF_k$. Thus, M_k is Poisson distributed with parameter $\kappa_k E[\hat{S}_k]$. To calculate the average completion time $E[\hat{C}_k]$ and the squared coefficient of variation $C^2[\hat{C}_k]$ the average number of failures $E[M_k]$ and the corresponding variance $\text{Var}[M_k]$ is needed. These are given by

$$E[M_k] = \kappa_k E[\hat{S}_k] \quad (3.24)$$

$$E[M_k^2] = \kappa_k^2 E[\hat{S}_k^2] + \kappa_k E[\hat{S}_k] \quad (3.25)$$

$$\text{Var}[M_k] = \kappa_k^2 E[\hat{S}_k^2] + \kappa_k E[\hat{S}_k] - (\kappa_k E[\hat{S}_k])^2. \quad (3.26)$$

Using Wald's law (see [Mül91]) the mean completion time is given by

$$\begin{aligned}
 E[\hat{C}_k] &= E[\hat{S}_k] + E\left[\sum_{\nu=1}^M D_{k,\nu}\right] \\
 &= E[\hat{S}_k] + E[M] E[D_k] = E[\hat{S}_k] + \kappa_k E[\hat{S}_k] E[D_k] \\
 &= \frac{E[\hat{S}_k]}{r_k} .
 \end{aligned} \tag{3.27}$$

Similar to this the variance can be evaluated by

$$\begin{aligned}
 \text{Var}[\hat{C}_k] &= \text{Var}[\hat{S}_k] + \text{Var}\left[\sum_{\nu=1}^M D_{k,\nu}\right] \\
 &= \text{Var}[\hat{S}_k] + E[M] \text{Var}[D_k] + \text{Var}[M] E[D_k]^2 \\
 &= \text{Var}[S_k] + \kappa_k E[\hat{S}_k] \text{Var}[D_k] + \\
 &\quad (\kappa^2 E[\hat{S}_k^2] + \kappa_k E[\hat{S}_k] - (\kappa_k E[\hat{S}_k])^2) E[D_k]^2 .
 \end{aligned} \tag{3.28}$$

Further conversions yield the form for the squared coefficient of variation of the completion time

$$C^2[\hat{C}_k] = C^2[\hat{S}_k] + \frac{r_k(1 - r_k) E[D_k](1 + C^2[D_k])}{E[\hat{S}_k]} . \tag{3.29}$$

3.11. Batch Processing

Batch processing is modeled on work center level. That means all process steps at a work center need to have the same batch size, i. e. the (b, b) batch rule is applied for all process steps at a work center. Different work centers can have different batch sizes, though. After processing a batch at a work center it is moved to the next process step as a batch. The transport batch size is therefore equal to predecessors's batch size³. Thus, different flows of batches having different batch sizes can arrive at the input of a work center. To investigate the flow of batches, first, the mean and SCV of the input batch size X_k have to be determined. The number of visits e_m at process step a_m (defined in equation (3.7)) applies for single parts, not full batches.

³The modeling of transportation using different transport batch sizes is discussed in section 7.2.6.

To get the number of visits referring to batch size $b_{w(m)}$ of process step a_m the value has to be divided by the batch size, formally

$$e_m^* = \frac{e_m}{b_{w(m)}}. \quad (3.30)$$

Using this the mean input batch size $E[X_k]$ at work center w_k can be aggregated by

$$\begin{aligned} E[X_k] &= \frac{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u e_m^* b_{w(m)} p_{m,n}}{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u e_m^* p_{m,n}} \\ &= \frac{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u e_m p_{m,n}}{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u \frac{e_m}{b_{w(m)}} p_{m,n}}. \end{aligned} \quad (3.31)$$

A similar result for the second moment of the input batch size can be achieved by

$$\begin{aligned} E[X_k^2] &= \frac{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u e_m^* b_{w(m)} p_{m,n}}{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u e_m^* p_{m,n}} \\ &= \frac{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u e_m b_{w(m)} p_{m,n}}{\sum_{u \in \mathcal{U}} \sum_{a_m \in \mathcal{A}_u} \sum_{a_n \in \mathcal{A}_{m,w_k}} \alpha_u \frac{e_m}{b_{w(m)}} p_{m,n}}. \end{aligned} \quad (3.32)$$

This enables the calculation of the squared coefficient of the input batch size by

$$C^2[X_k] = \frac{E[X_k^2]}{E[X_k]^2} - 1. \quad (3.33)$$

The arrival rate of batches at work center w_k is the reciprocal of the inter-arrival times of process batches at the work center. Having calculated the work centers' number of visits \hat{e}_k the arrival rate of batches in respect to batch size X_k is given by

$$\lambda_k^X = \frac{\lambda \hat{e}_k}{E[X_k]}. \quad (3.34)$$

This value is exact because of the balance of flows in the network. The SCV of the inter-arrival time of batches cannot be derived in this manner, but has to be calculated on basis of the network decomposition approach presented in the next section.

3.12. Network Decomposition Method

Exact solutions for queueing networks can only be found for Markovian systems, so called Jackson networks. First, Burke [Bur56] proved that the output of a $M/M/c$ system is once again a Poisson process with the exact same rate. This enabled the isolated treatment of nodes in the system like it is proposed by Jackson [Jac57]. Due to the lack of exact solutions for more general networks, approximation schemes have been developed. The decomposition method used in this model is similar to the one presented by Kobayashi [RK73], Bitran [BT88, BT89], Gelenbe [Gel75], Pujolle and Ai [PA86], or Whitt [Whi83], for example. Compared to these models, Zisgen's approach [Zis99] enables the creation of more general networks concerning batch processing. In Bitran and Tirupati's model either batch processing or batch arrivals are permitted. In order to build general models alternating stations have to be used. The servers are restricted to single stations. This restriction does not exist in Hanschke's model [Han95] which permits batch processing and batch arrivals at multi-server systems. The batches are split after processing and routed to the following nodes in the network, thus, bulk arrivals are not permitted. Zisgen extends this model to incorporate transportation of batches and thus bulk arrivals. Formally, general networks of $G^X/G(b, b)/c$ stations can be analyzed.

Network decomposition approximation approaches separate all work centers in the network to isolated nodes which can then be analyzed using the known formulae. These methods which are based on the assumptions

1. the nodes of the network can be treated as stochastically independent and
2. two-parameter approximations (typically the mean and variance) provide reasonable and accurate results

have now become a standard and well-accepted technique at the network level [CFY96].

To separate the work centers it is assumed that the output of work centers are renewal processes. This is only the case, if work centers are $M/M/1$ stations or if the utilization of a work center is near to one (see [Kin64]). It can be shown that even if the assumptions are not fulfilled reasonably good approximations for the general case can be found (see [Zis99], [RK73] and [CL81]). Moreover, in many practical scenarios the *high traffic* case with utilizations near to one is most important.

To decompose a queuing network typically systems of linear equations for the means and SCVs of the inter-arrival times at each work center need to be solved. The first system to calculate the means has already been solved by calculating the number of visits at each work center. This information can be used to derive the mean inter-arrival time of batches at work centers. Because of the flow balance no approximation is needed in that case. During the course of this section the second system needed to calculate the SCVs of inter-arrival times is derived.

Figure 3.2 shows the streams of parts at work center w_k which have to be investigated in order to derive a system of equations for the SCVs of the inter-arrival times of batches at the node. The task which is complicated due to potential batch processing at arbitrary network nodes is split into four steps:

1. *Modulation of the input stream.* In section 3.11 merging batches to the input batch size X_k has been described. The resulting input stream I_k^X is then modulated at the virtual batching station in front of work center w_k in order to form the process I_k^b of batches with process batch size b_k .
2. *Approximation of the output stream.* As the output of a work center can also be the input of successive nodes in the network the output stream D_k^b of batches of size b is of special interest. At this step an equation showing the SCV of the inter-departure times in terms of the SCV of the inter-arrival times is developed.
3. *Splitting of the output stream with respect to the specified transition probabilities.* After the stream D_k^b has been determined it has to be split up according to the transition probabilities $(\hat{p}_{i,k})_{i,k=1,\dots,W}$ specifying the probability that a batch moves from node w_i to w_k .
4. *Overlay of multiple output streams to a single input stream.* Finally, the effects of multiple output streams with different batch sizes on the variance of the resulting input stream have to be evaluated. The result will be an equation showing the SCV of the input stream $C^2[\hat{I}_k]$ in terms of the SCV of the output stream.

At each step a part of the final system of linear equations is developed. The outcome of the first step is used in the second step, step three is needed for step four. Finally, the results of steps two and four can be combined in

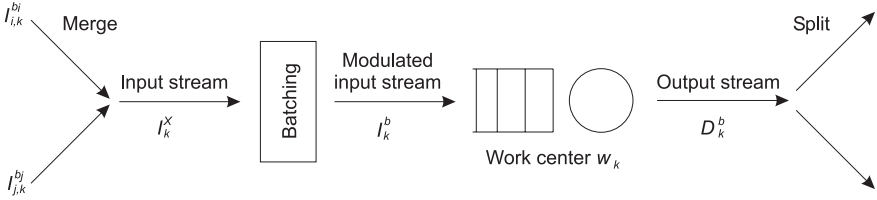


Figure 3.2.: Streams used in the network decomposition approach

order to construct a system of linear equations enabling the approximation of the queueing network in closed form.

3.13. Performance Measures

The utilization of a work center can be given either as the net or the total utilization. The first is calculated using the mean service time of a process step, for the latter the mean completion time is used. Thus, the net utilization does not take any machine failures into consideration. It is given by

$$\varrho_k^{net} = \frac{\lambda \hat{e}_k E[\hat{S}_k]}{c_k b_k} \quad \text{for } k = 1, \dots, W. \quad (3.35)$$

The total utilization can similarly be computed by

$$\varrho_k^{tot} = \frac{\lambda \hat{e}_k E[\hat{C}_k]}{c_k b_k} = \frac{1}{r_k} \varrho_k^{net} \quad \text{for } k = 1, \dots, W. \quad (3.36)$$

The net utilization is based on the normally unrealistic assumption that no machine failures can occur. Its main purpose is therefore to show, in comparison with the total utilization, potential reserves in capacity which could be possibly exploited by deploying a more sophisticated maintenance schema. From now on the utilization ϱ_k of work center w_k therefore always refers to the more realistic total utilization.

The maximum arrival rate λ^{max} of the queueing network is determined by the bottleneck which limits the flow of material through the network. It is given by

$$\lambda^{max} = \min_{k=1, \dots, W} \left\{ \frac{c_k b_k}{\hat{e}_k E[\hat{S}_k]} \right\} = \min_{k=1, \dots, W} \{ \varrho_k^{tot} \}. \quad (3.37)$$

The maximum arrival rate per product type can be derived from the overall maximum arrival rate with the help of the product mix by $\lambda_l^{max} = \alpha_l \lambda^{max}$ for $l = 1, \dots, U$.

The utilization of the queuing network determines, if the performance measures which are presented in the following section can be calculated or not. If the total utilization exceeds 1.0 the network is overloaded and no stationary state can be reached. No work-in-process can be calculated in this situation. The intermediate results, e. g. the utilization of the bottleneck work centers can be derived, though. This is a clear advantage of the queuing network approach as all capacity related statements can be made even if the network is overloaded.

3.14. Work-in-Process and Lead Time

The queue length and the number of parts in the system are the central performance values of a queuing system as many other important performance measures can be derived from them by Little's law [Lit61]. Different approaches for approximations of $G^X/G(b, b)/c$ have been developed. This model uses the approximation introduced by Allen and Cunneen. First, the probability that all c_k servers of the work center are busy is approximated by Erlang's c-formula⁴

$$P_c = \frac{\frac{(c\varrho)^c}{c!} \frac{1}{1-\varrho}}{\sum_{i=1}^{c-1} \frac{(c\varrho)^i}{i!} + \left(\frac{(c\varrho)^c}{c!} \frac{1}{1-\varrho} \right)}. \quad (3.38)$$

This is used to approximate the mean queue length at the work center by

$$E[\hat{Q}_k] = \frac{b_k - 1}{2} + \frac{\varrho_k}{1 - \varrho_k} P_c \frac{b_k \left(C^2[\hat{I}_k] + C^2[\hat{C}_k] \right)}{2} \quad \text{for } k = 1, \dots, W. \quad (3.39)$$

The mean waiting time which is the time that parts have to wait in the queue in front of a work center is derived using Little's law

$$E[\hat{W}_k] = \frac{E[\hat{Q}_k]}{\lambda \hat{e}_k} \quad \text{for } k = 1, \dots, W. \quad (3.40)$$

⁴For better readability the indices of the work center have been omitted as this formula applies only to one work center.

The expected lead time at a work center is the sum of the mean waiting time and the average completion time

$$E[\hat{L}_k] = E[\hat{W}_k] + E[\hat{C}_k] \quad \text{for } k = 1, \dots, W. \quad (3.41)$$

The work-in-process (WIP) at a work center $E[N_k]$ is the number of parts waiting in its queue plus the parts that are currently being processed. This can be calculated by

$$E[\hat{N}_k] = E[\hat{Q}_k] + \varrho_k c_k b_k \quad \text{for } k = 1, \dots, W. \quad (3.42)$$

3.15. Overall Performance Measures

Once the performance measures of the work centers have been calculated they can be aggregated in order to obtain values for the whole model. The overall work-in-process is the sum of all parts at all work centers

$$E[N] = \sum_{k=1}^W E[\hat{N}_k]. \quad (3.43)$$

To calculate the overall expected lead time the number of visits at the work centers have to be taken into consideration

$$E[L] = \sum_{k=1}^W \hat{e}_k E[\hat{L}_k]. \quad (3.44)$$

The raw process cycle time is defined to be the time needed to produce one complete part without having to wait for processing. It includes rework and yield, but omits machine failures. The calculation is similar to the one of the overall lead time

$$E[T] = \sum_{k=1}^W \hat{e}_k E[\hat{S}_k]. \quad (3.45)$$

Finally, an efficiency measure in terms of the time needed to produce parts can be established by comparing the raw process cycle time to the overall lead time. The measure is defined by

$$\varepsilon = \frac{E[T]}{E[L]}. \quad (3.46)$$

3.16. Performance Measures for Process Steps and Product Types

All of the performance measures derived in the previous section apply for work centers. For a more detailed analysis these values are desired for process steps and product types, as well. The expected lead time of process step a_m is determined by the waiting time in front of the work center plus the mean completion time of the process step

$$E[\tilde{L}_m] = E[\hat{Q}_{w(m)}] + \frac{E[S_m]}{r_{w(m)}} \quad \text{for } m = 1, \dots, A. \quad (3.47)$$

The mean queue length of parts waiting in front of a work center for a specific process step can be calculated by

$$E[\tilde{Q}_m] = E[\hat{Q}_{w(m)}] \frac{e_m}{\hat{e}_{w(m)}} \quad \text{for } m = 1, \dots, A. \quad (3.48)$$

The queue length of a specific product type at a work center is determined by

$$E[\tilde{Q}_{l,k}] = E[\hat{Q}_k] \frac{\hat{e}_{l,k}}{\hat{e}_k} \quad \text{for } l = 1, \dots, U \text{ and } k = 1, \dots, W. \quad (3.49)$$

The expected number of parts being processed or waiting in front of a work center w_k of product type u_l can be expressed by

$$E[\tilde{N}_{l,k}] = E[\tilde{Q}_{l,k}] + \frac{\hat{e}_{l,k}}{\hat{e}_k} \varrho_k c_k b_k \quad \text{for } l = 1, \dots, U \text{ and } k = 1, \dots, W. \quad (3.50)$$

The same applies to the lead time of a product at a work center using

$$E[\tilde{L}_{l,k}] = \frac{E[\tilde{N}_{l,k}]}{\lambda \alpha_l \hat{e}_{l,k}} \quad \text{for } l = 1, \dots, U \text{ and } k = 1, \dots, W. \quad (3.51)$$

The lead time of a single product type over all work centers is calculated by

$$E[\tilde{L}_l] = \sum_{k=1}^W \hat{e}_{l,k} E[\tilde{L}_{l,k}] \quad \text{for } l = 1, \dots, U. \quad (3.52)$$

The expected work-in-process of product type l in the whole queuing model is determined by

$$E[\tilde{N}_l] = \sum_{k=1}^W E[\tilde{N}_{l,k}] \quad \text{for } l = 1, \dots, U. \quad (3.53)$$

The expected raw process cycle time of a product type is the mean completion time of all process steps of a product type weighted by the process steps' number of visits:

$$E[\tilde{T}_l] = \sum_{a_m \in A_l} e_m E[S_m] \quad \text{for } l = 1, \dots, U. \quad (3.54)$$

3.17. Remaining Lead Time

The average lead time $E[L]$ defined in equation (3.44) is the expected value of the lead time of all parts which are started into the network whether they become good parts or scrap. In practical scenarios often the lead time of good parts only is needed. In a queueing network with positive scrap rates the scrap parts might leave the network much earlier than the finished parts distorting the overall expected lead time.

Moreover, sometimes the expected remaining lead time of a good part at process step a_m is needed. To calculate these performance measures another system of linear equations has to be solved. (The probability that a part at a_m becomes a good part has already been calculated in 3.7.)

For product type u_l the remaining lead time of process step a_m is recursively defined by

$$E[\check{L}_m] = \sum_{n=1}^A p_{m,n} \left(E[\check{L}_n] + g_n E[\check{L}_m] \right) \quad \text{for } m = 1, \dots, A. \quad (3.55)$$

This is the weighted sum of all successor's remaining lead times plus the process steps' own lead time $E[\check{L}_m]$ which can be interpreted as the transition time between the process step and its successor multiplied by the probability that a unit is not scrapped g_n after the successor. Rewriting (3.55) as

$$E[\check{L}_m] = \sum_{n=1}^A p_{m,n} E[\check{L}_n] + E \left[\check{L}_m \sum_{n=1}^A p_{m,n} g_n \right] \quad \text{for } m = 1, \dots, A \quad (3.56)$$

leads to the matrix form $\check{l} = P\check{l} + z$ where

$$\check{l} = \begin{bmatrix} \check{l}_1 \\ \check{l}_2 \\ \vdots \\ \check{l}_A \end{bmatrix}, \quad P = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,A} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,A} \\ \vdots & \vdots & \ddots & \vdots \\ P_{A,1} & P_{A,2} & \cdots & P_{A,A} \end{bmatrix}, \quad z = \begin{bmatrix} E[\check{L}_m] \sum_{n=1}^A p_{1,n} g_n \\ E[\check{L}_m] \sum_{n=1}^A p_{2,n} g_n \\ \vdots \\ E[\check{L}_m] \sum_{n=1}^A p_{A,n} g_n \end{bmatrix}. \quad (3.57)$$

The remaining lead time of good parts at process step a_m is given by $E[\check{L}_m]/g_m$. The overall lead time of good parts through the network can be obtained as the weighted sum of each product type's remaining lead time of good parts at the corresponding source process steps

$$E[\check{L}] = \sum_{l=1}^U \alpha_l \frac{E[\check{L}_{\text{source}(l)}]}{g_{\text{source}(l)}}. \quad (3.58)$$

Part II

EPOS - A System for Integrated Simulation

Chapter 4

System Architecture

4.1. Introduction

EPOS (Enterprise Production Planning and Optimization System) is a prototype that implements most of the requirements of *integrated simulation* and can be used to incorporate the advantages of state-of-the-art simulation methods into the business processes of production planning. EPOS integrates existing shop-floor-control systems, automatic model generation, and a user-centric presentation of the simulation results in order to create a powerful, distributed, scalable, and secure decision support system.

The architecture of a system for *integrated simulation* will go far beyond a single desktop application. The reason for this lies in the complex requirements that the model of *integrated simulation* demands of an actual system. These conceptual requirements lead to technical challenges. An example might be the principle of the distributed parameter input which brings up the question how the system can support parameter updates by many users at different production sites in a secure and efficient way.

As EPOS is designed to make use of a company's infrastructure and information assets, it will have to be integrated into an IT landscape that is likely to be very heterogeneous. The emerging e-business tools and today's middleware architectures like JDBC and CORBA, as well as the use of standard (internet) protocols make it possible to create such distributed

and integrated systems efficiently.

Another important point to keep in mind when creating these kinds of systems is *scalability*. The rising complexity of production processes — especially in semi-conductor manufacturing — leads to simulation models that tend to be very large and complex¹. These models and a large number of persons and production-sites which are even likely to increase during the operation must be manageable in an efficient way. But scalability not only means that the system has to cope with increasing requirements, but also that one should be able to run the system on a much smaller set-up than the complete enterprise system. For example, the *EPOS Analyzer* — the interactive simulation client of the EPOS-System (see section 6.2) — can run together with the simulation server on a standard Laptop under the Linux operating system.

As the requirements defined in the model of *integrated simulation* are rather demanding, specialized tools are incorporated into the EPOS system. The most suitable applications are chosen for the subsystems, e. g.

- a relational database for the collection and retrieval of the well-structured plan parameters,
- a Lotus/Notes database for the administration of semi-structured documents,
- Java applets for user front-ends as they allow an efficient roll-out of the application,
- a C++ queueing network evaluator that allows a fast evaluation of a queueing network's performance measures
- commercially available report generators that provide means for a quick definition and convenient scheduling of HTML reports from relational databases,
- MS Access for an administrator front-end that needs no control of row-specific access right on database tables.

¹The simulation of the Mainz wafer-line consists of more than 15.000 objects. These can be work centers, products, operations, routings, etc.

All these components communicate via standard protocols like HTTP, CORBA, JDBC, ODBC so that additional subsystems for answering further questions can easily be incorporated into the system, i. e. EPOS is an open system.

4.2. System Overview

As a system for *integrated simulation* EPOS makes use of many different subsystems communicating over the company's intranet by standard protocols. This chapter describes EPOS by explaining its subsystems and the underlying data structures and dynamic transactions.

The stereotyped deployment diagram in figure 4.1 shows the subsystems (hardware nodes) of EPOS and their communication links (interfaces). The stereotyped nodes are:

- *Database server.* This is the central data warehouse that collects all planning parameters, either imported from shop-floor-control systems or manually maintained, i. e. the master files, and the simulation results (see section 4.4).
- *Model generator.* The model generator generates the real-world model from the data warehouse, transforms it into a model for a specific simulator, and stores the simulation results after the model has been evaluated by the simulation server. Automatic model generation is described in section 7.2.3.
- *Simulation server.* The simulation server provides the queuing theory formulae. It allows to create simulation models, evaluates them and offers the results in its API (see section 6.1).
- *Tool-parameter-sheets client.* All data that has to be entered manually by the responsible persons is collected by this Java applet. It is embedded in a Lotus Notes document so that it gets distributed by the Notes server.
- *HTTP server.* The HTTP Server is used for the publication of static HTML reports, the generation of dynamic HTML reports, and can also be used for the distribution of the tool-parameter-sheet client, if Notes is not available.

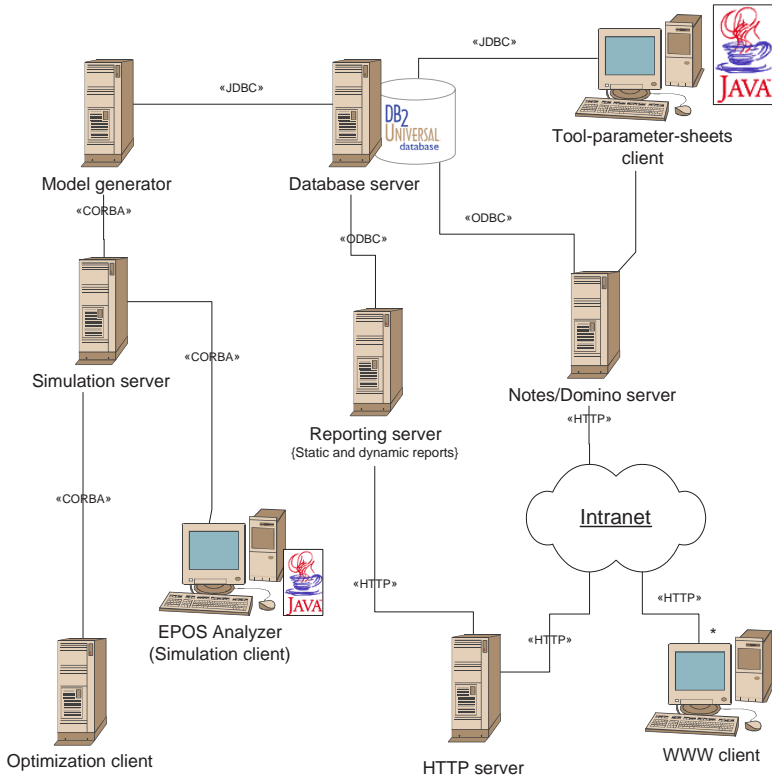


Figure 4.1.: Overview of the EPOS architecture

- *Notes/Domino server.* The Notes server provides the EPOS Notes Database. It contains detailed views of the tool-parameter-sheets and additional information on EPOS like help documents and administrative information (data definitions, SQL scripts of the central plan parameter database). By using Notes Domino features the EPOS database is published as the EPOS intranet web site.
- *EPOS Analyzer.* The EPOS Analyzer is a graphical user interface for the simulation server. It allows to create, modify, delete, and simulate models on the simulation server (see section 6.2).

- *Optimization client.* The optimization client provides sophisticated methods for solving different optimization tasks in the production environment, e.g. it allows to determine optimal product mixes, optimal routings etc. (see section 10).
- *WWW client.* The reports of actual data and simulation results are provided as HTML pages that are hosted on an intranet server. Moreover, the Notes database can be accessed via a web client as well as its contents is distributed by a Notes/domino server.
- *Reporting server.* Two different kinds of reporting servers are used within EPOS: static HTML pages and dynamically created pages (see chapter 8).

The subsystems have to be able to communicate with each other as shown in the deployment diagram. The communication links between the nodes are established by:

- *CORBA.* The common object request broker architecture manages objects for client/server applications distributed in a network. CORBA allows programming on the object level hiding tedious socket programming, the development of proprietary protocols, and data conversion between different hardware platforms (big endian, little endian machines, etc.).
- *HTTP.* The hypertext transfer protocol enables the transfer of plain text with special mark-ups and hyper-links (HTML) and other binary data files from a web-server to a web client (browser). EPOS uses HTTP for the transmission of static and dynamic reports of simulation results and the EPOS intranet site.
- *ODBC.* Open database connectivity is a middleware defined by Microsoft that allows to access remote database servers in a network.
- *JDBC.* Java database connectivity is a middleware for Java programs that allows to access remote database servers in a network.
- *Notes.* The work group system Lotus Notes uses its own protocol to provide the client with database documents from a server. This protocol is also able to provide end-users with Java applets embedded in Notes documents.

All these subsystems work on the same database, the central EPOS data warehouse. How the master data is organized is shown in the next section.

4.3. Core Components

In this section the functionality of EPOS is modeled by object-oriented methods. The main classes, their relationships, and their interactions are shown in detail. For clarity, the theoretical explanations are accompanied by an example whose details and relational database tables can be found in appendix A.

The UML diagrams in this section present the logical view on the artifacts of the system like classes and their static and/or dynamic interactions. In the following no distinction is made between classes and objects if it is clear from the context what is meant. Thus cluttering up the text with too many technical details is avoided. Only if the distinction is necessary, it is stated explicitly. Unless stated differently, the UML diagrams in the following sections are class diagrams, i. e. the class **Company** can be instantiated by different companies from the real world.

4.3.1. Company Structure

The most important part of EPOS is its model of the company structure as shown in figure 4.2. Starting at the top of the hierarchy the class **Model** serves as a container for several companies that are to be modeled and simulated. The class **Model** allows to separate the space of all companies into several sections. For each model different assumptions may hold that can be reflected during the modeling and simulation phase.

The class **Company** corresponds directly to the business organizations found all over the world. Although many companies are running totally different productions in different parts of the world, there are still some characteristics of a company that have to be defined at this level. These are modeled as attributes of the **Company** class like a company's name, divisions, and product forms .

Large companies operating worldwide are often divided into several divisions. These divisions reflect different business fields, sub-companies, or regional organizations and are modeled by the class **Division**. Divisions can be subdivided into locations. These normally correspond to production sites of a company.

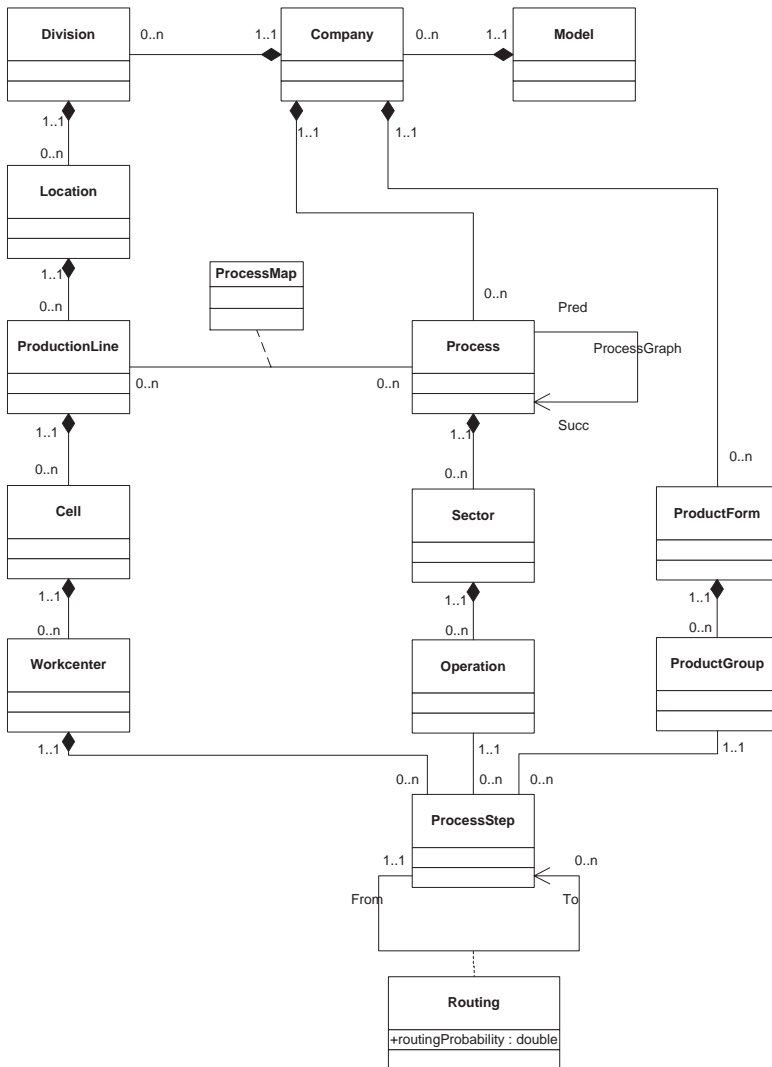


Figure 4.2.: Company structure including the classes work center, product, and operation

At a location typically several products are manufactured in different production lines. The production lines play an important role for the simulation as they are the entities that are modeled and simulated by the EPOS system. Thus it is impossible to carry out separate simulations for the next smaller class **Cell** in the hierarchy. The class **Cell** allows to partition a production line. The production line of the example in appendix A is divided into the cells *Lap*, *Sputter*, *Photo*, *Plate*, and *Test*. Each cell contains a set of work centers, e.g. the photo cell contains the steppers *Stepper 0815* or *Stepper 4711*.

The classes from **Division** to **Workcenter** are the first hierarchical dimension. The UML diagram shows the hierarchy as a composition: Each class is dependent on the class above it in the hierarchy, i.e. work centers cannot exist without a containing cell, the cells cannot exist without a corresponding production line and so on.

The next hierarchy starts with the class **Process**. Processes describe the actions that are necessary for manufacturing products on a rather abstract level. Processes are arranged in a graph, as for multi-level product structures each intermediate product has a process for its production. The figure shows the successor/predecessor relationship (process graph) for intermediate products. In a production line several intermediate products can be produced each having its own process. Simultaneously, a process is a rather abstract description of how to manufacture a product so that a process can be performed in different production lines. Therefore there is a many-to-many association between production lines and processes. Technically, a many-to-many relationship can be realized by a table shown by the association class **ProcessMap**².

Processes are composed of sectors containing operations. An operation is the smallest step in the manufacturing of a product. An operation is independent of a work center and a product as it just describes what has to be done without specifying the resource. Thus, the most important attribute of an operation is its name.

The third dimension contains information on products. As products are considered to be unique within a company, different instances of the class **ProductForm** represent different categories of the company's products. A complete discussion of all aspects concerning the products is given in section 4.3.3.

²This corresponds exactly to the table design of the underlying relational database.

Product groups are the entities that are modeled as products in the simulation. Thus it is possible to group technically identical products so that the redundant specification of simulation details is avoided. Often a product sold to different customers or offered on different markets gets a new name for each occasion. But as the model of *integrated simulation* suggests, redundancy has to be avoided under any circumstances in order to get maintainable simulation models.

The data structure described in this chapter is a typical snowflake structure found in data warehouses and on-line analytical processing (OLAP). The term on-line analytical processing was developed to distinguish data warehousing activities from "On-Line Transaction Processing" - the use of computers to run the ongoing operation of a business. OLAP is the most widely used term for multi-dimensional analysis software. In its broadest usage the term OLAP is used as a synonym of *data warehousing*. In a more narrow usage, the term OLAP is used to refer to the tools used for multi-dimensional analysis (see [HS00, Inm99, Inm00]).

The snowflake scheme of the company structure is formed by the three dimensions, namely the work centers, product groups, and the operations. They are associated to the class `ProcessStep`. Each dimension can be found at the bottom of a hierarchy. The hierarchies of the three dimensions allow the typical OLAP operations like drill-down and drill-up, i.e. the data of process steps can be viewed at different aggregation levels. Descending the hierarchy from top to bottom constitutes the drill-down operations, the view of the data is changed to a greater level of detail. Aggregation on higher levels of the hierarchy is allowed by the drill-up operations, the view of the data is changed to a higher level of aggregation. As all steps in the hierarchy are one-to-many relationships, each of the three dimensions is a tree.

Process steps belong to work centers (composition). But they need to be instantiated with an operation and a product group. The combination of these three entities determines the parameter values of the process step, like a cycle time and an actual load size. For example, the tool *Stepper 4711* performs the operation *0150 Expose 1* for the products *Lambda* and *Sigma*. Thus, the parameters cycle time and actual load size can be specified as shown in table 4.1.

Very important for the correct production is the order in which operations are performed at work centers. This is given by the process flow which is a graph whose vertices are the process steps. A process step might have several successors for three reasons:

Operation	Product	Cycle time	Act. load size
0150 Expose 1	Lambda	5 min	1 Wafer
0150 Expose 1	Sigma	5 min	1 Wafer

Table 4.1.: Sample process steps for *Stepper 4711*

1. *Rework.* When an operation for a part has been finished, the part has to be moved to the next operation. This can either be the next operation in the process flow or — in the case of rework — a previous or a special rework operation. Rework normally occurs at test operations after measurements or inspections to decide which parts do not fulfill the quality requirements.
2. *Junctions.* If the next process step for a part can be performed at different work centers, the process flow branches. Routing probabilities specify the long-term averages of the way the products take.
3. Rework and junctions can be found simultaneously at a process step.

Each edge of the process graph contains a routing probability defined by the attribute of the association class **Routing**. In the case of rework, the routing probability is used to model the yield of an operation. The first time yield (FTY) y is the routing probability to the next process step. Let s be the scrap ratio (in percent) at the operation. Then the routing probability $p = 1 - y - s$ is assigned to the edge that leads to the rework operation (see section 7.2.5).

4.3.2. Work Centers

A work center represents a set of several identical tools. The term *identical* in this definition refers to both the hardware and the use of the tools. This means that identical tools that are dedicated to the production of different products do not belong to the same work center. Work centers are grouped into the cells of a production line as shown in figure 4.2.

A work center has many attributes for its administration as shown in table 4.2, for example, special fields keep track of the last editor and the time when the modifications took place. The set of persons responsible for the parameters of a work center can be divided into four groups or departments:

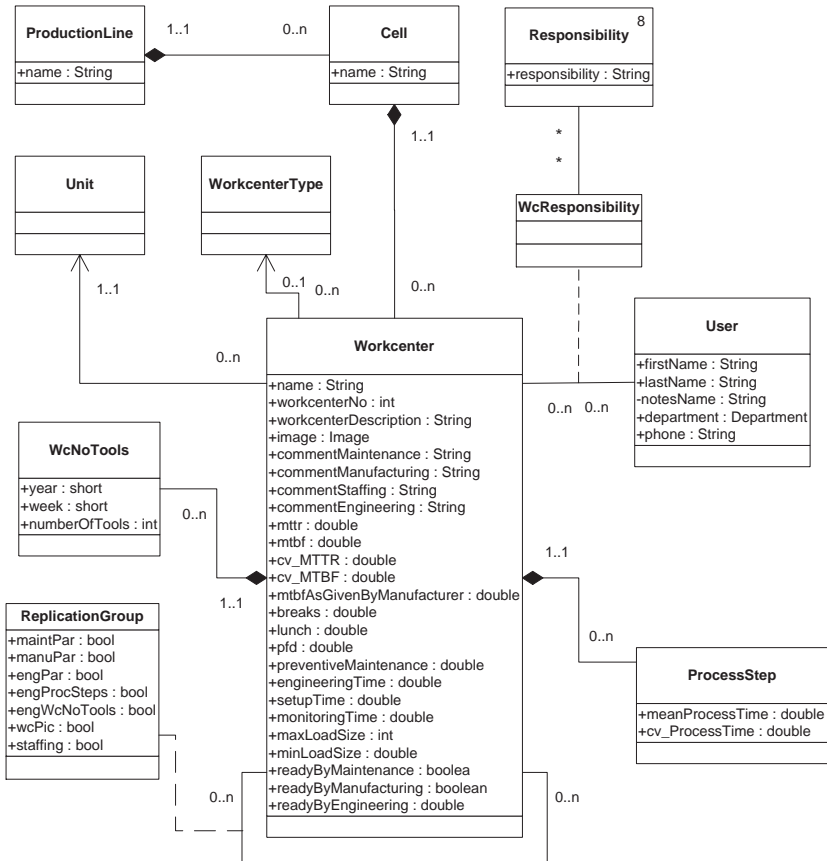


Figure 4.3.: Work centers

- *Maintenance.* The maintenance department is responsible for keeping the tools in order and to repair them if they are out of order. Additionally, preventive maintenance might be necessary. During these maintenance periods the machine is not available for production.
- *Manufacturing.* The manufacturing department is responsible for the

Attribute	Comment
workcenterNo	the unique identifier of a work center
name	the name of the workcenter
location	shop-floor coordinate string of work center
dirty	flag indicating that data has been changed (see chapter 5)
unit	the default unit of the products processed at the workcenter
workCenterType	a type indicating additionally needed parameters (see section 4.3.2)
workcenterVariance	numerical value in $[0, \dots, 100]$ that is mapped to a coefficient of variation
planWorkcenter	a workcenter providing plan parameters (see section 4.3.2)
lastEdEng	time-stamps showing time of the last edit for every section
lastEdMaint	
lastEdManu	
lastEdByEng	last persons who edited data for every section
lastEdByMaint	
lastEdByManu	
readyByEng	flag showing that the responsible person has completed his input, see chapter 5 for a detailed description of the complete business process
readyByMaint	
readyByManu	

Table 4.2.: Attributes of class **Workcenter** for administration

operators, their breaks, etc. and has to assure that the products are manufactured on time.

- *Engineering.* The engineering department is responsible for the production process and the technology of the products. It carries out experiments to examine how the process can be improved. Its duty is to specify the parameters of the process steps.
- *Staffing.* The planning of manpower assignment is often placed under

Attribute	Comment
breaks	operator breaks [h/day]
lunch	operators' lunch times if machine does not continue to work [h/day]
pdf	personal fatigue and delay [h/day]
engineering	engineering time [h/week]
setup	set-up time [h/week]
monitoring	monitoring time [h/week]
prevmaint	time for preventive maintenance [h/week]
mtbf	mean time between failures
mtbfManufact	mean time between failures as given by manufacturer
mttr	mean time to repair
cvmtbf	coefficient of variation for time between failures
cvmttr	coefficient of variation for time to repair

Table 4.3.: Attributes of class **Workcenter** for reliability calculation

the responsibility of manufacturing.

Moreover, the class **Workcenter** contains attributes for comments for each section (*Manufacturing*, *Engineering*, *Staffing*, and *Maintenance*) and one field for an overall work center comment filled by the administrator.

The work centers of a production line are not always available for production, for example, if a tool is in maintenance or if no operators are available. In order to take the down times of the work centers into consideration the attributes shown in table 4.3 are to be filled by the three departments. As the attributes for tool down times are given in the units the responsible persons are accustomed to (see table 4.3), the reliability r of a work center can be calculated as

$$\begin{aligned}
 r = 24 & - \left(\frac{24}{mtbf + mttr} \cdot mttr \right) \\
 & - \left(\frac{prevmaint + setup + engineering + monitoring}{7} \right) \\
 & - lunch - breaks - pdf.
 \end{aligned}$$

The remaining attributes of class **Workcenter** shown in table 4.4 allow to specify technological characteristics of the work centers, like the load size

Attribute	Comment
serviceType	possible values: serial or parallel
loadsizeMax	l_{\max} maximum number of wafers that can be loaded into a tool
loadsizeMin	l_{\min} minimum number of wafers that can be loaded into a tool
opSetup	constant time for each operation (in-claim, out-claim, set-up)
opTimeConst opTimeVar	constant time an operator is needed for an operation percentage of the machine's cycle time an operator is needed

Table 4.4.: Attributes of class **Workcenter** for cycle time calculation

and the type of operation, or are used to calculate the head count needed (see section 4.3.6). The service type of a work center specifies if production parts are processed in parallel or not. Note that to define this parameter the process in a tool is considered but not the way a tool is loaded. Some tools can be loaded with a batch of parts, however, in the machine each part is taken and processed in turn. These tools are considered to be serial tools.

The cycle time given as the attribute **meanProcessTime** of class **ProcessStep** is related to the actual load size. Thus, for smaller or larger load sizes $l, l_{\min} \leq l \leq l_{\max}$, the cycle time of a process step $c(l)$ can be calculated depending on the service type as follows

$$c(l) = \begin{cases} opSetup + l \cdot \frac{c_p - opSetup}{l_{act}} & \text{if } \text{serviceType} = \text{serial} \\ c_p & \text{if } \text{serviceType} = \text{parallel} \end{cases} \quad (4.1)$$

where c_p denotes the cycle time given in the process step. Note that equation 4.1 is a linear approximation for most cases. Figure 4.4 shows a plot of the function $c(l)$.

The number of tools in a work center can change over time. The class **Workcenter** keeps track of those changes by maintaining a list of objects of class **WcNoTools** each representing the time of a change and the new number of tools. For example, two entries (*year, week, number*) like (1999, 10, 2) and (1999, 40, 3) show that since week 10 in the year 1999 there had been 2 tools in the work center and that the number of tools increased to 3 in week 40 in 1999.

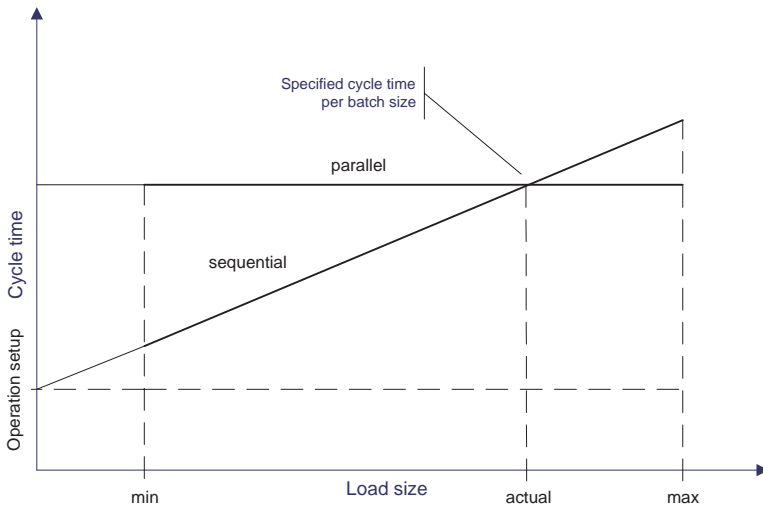


Figure 4.4.: Cycle time as a function of load size

Following the model of *integrated simulation* the parameter values for a **Workcenter** have to be maintained by the persons responsible. To support the process described in chapter 5 the responsibilities have to be defined. The class **Responsibility** provides the eight basic responsibilities *Maintenance*, *Manufacturing*, *Engineering*, *Staffing*, *Engineering (Backup)*, *Maintenance (Backup)*, *Manufacturing (Backup)*, and *Staffing (Backup)*.

The users are defined with the help of the class **User** that collects the user's first and last name, department, phone number and Lotus Notes (or e-mail) address. The users are assigned to the work centers whereby the association class **WcResponsibility** defines which responsibility the user has. The many-to-many relationship between users and work centers shows that a user can have several responsibilities at several work centers as well as that several users can be responsible for the same work center.

Work Center Groupings

The UML diagram in figure 4.5 shows two important refinements for work centers, namely work center groups and work center types. These allow to

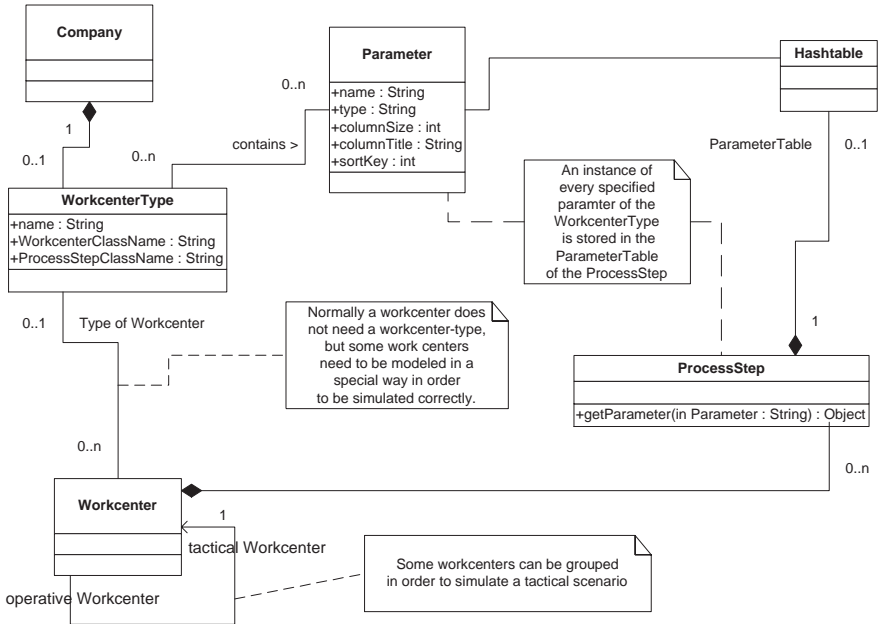


Figure 4.5.: Work center types and groupings

control the automatic model generation. Work center groups are a means to assign several operational work centers to a tactical work center. Thus simulation models for either operational or tactical planning can be generated. The operational work centers can reflect tool dedications that are automatically resolved when tactical simulation models are generated: The process steps of all operational work centers are united on a single work center whose number of tools is set to sum of the number of operational work centers. For example, a semi-conductor company has three steppers. In an operational setting, each stepper is a work center of its own and is set up to perform all operations necessary for a special product, i.e. each stepper is dedicated to a certain product. A tactical stepper combines the three steppers. If a tactical model is generated, a single stepper one work center is created with three tools and all the process steps are united on that work center.

The difference between the operational and the tactical scenario is that

in the operational setting three dedicated queues are to be found whereas in the tactical scenario a common queue is established that is served by three servers. The term *operational* refers to the fact that it is normally too expensive to qualify all identical tools for all products. Thus there is normally some dedication. In tactical planning these dedications should be neglected to achieve tight production plans. Note that tactical scenarios are supposed to perform better than operational ones with respect to capacity, work-in-process, etc.

Work Center Types

A company can contain special **WorkcenterTypes** that allow to add parameters to the process steps at certain work centers. For example, an inspection work center can be assigned a parameter *sample* that indicates the percentage of parts to be checked. The attributes **workcenterClassName** and **processStepClassName** indicate the name of the classes of the simulation server that allow the correct handling of the work center and the corresponding process steps, respectively (see section 7.2.3).

Each work center type can have several associated parameters, represented by the class **Parameter**. Its attributes are shown in table 4.5. An instance of class **ProcessStep** collects its parameters in a hash table. Thus all parameters specified for a process step can be accessed efficiently during the model generation by means of the method **getParameter**.

In an object-oriented view work center types can be realized by inheritance. The relationship of a work center and a typed work center is that of a super class and a subclass. Thus it is not necessary to specify a work center type, however, it is a means to provide a specialized implementation

Attribute	Comment
name	name of the parameter
type	type of the parameter, e. g. percent, double, boolean
columnSize	width of a column showing this parameter
columnTitle	title of a column showing this parameter
sortKey	order in which parameters of a work center are to be displayed

Table 4.5.: Attributes of class **Parameter**

of a work center. The UML diagram shows the aspects of implementing the inheritance relationship in a relational database.

Plan Work Centers

Plan work centers allow the administration of plan parameters for work centers. Figure 4.6 shows on the left-hand side the company structure presented in section 4.3.1 and on the right-hand side a similar hierarchy for plan work centers. The class `PlanData::WCGroup` allows to combine several plan work centers in a group. This group is distinct from production lines and locations as the same type of tool can be used in different production lines. Thus the parameters of the plan work centers allow to maintain company-wide plan parameters.

For example, the attribute `reliability` provides the desired value for a work center's reliability. The attribute `MTBF` provides the mean time between failures as given by the manufacturer of the tool or by the company's planners. This parameter can be compared with values taken from shop-floor-control systems.

4.3.3. Products

EPOS offers different views on products at two different aggregation levels as shown by the classes `Product` and `ProductGroup` in figure 4.7. On the one hand, a class is needed for the products sold to customers. For these products (instances of the class `Product`) a demand is entered into the volume plan. On the other hand, some products might be similar and can be modeled more efficiently as a single group product in the simulation. Each product group belongs to a company and is described by its form and its size:

1. **ProductForm.** The product form describes the physical shape of a product, e.g. a wafer, a row, or a slider. The form can be different from the `unit`, if a product group is an aggregation of several other product groups; for example, a wafer might consist of n sliders, thus in terms of a unit $1 [\text{wafer}] = n [\text{sliders}]$, but a wafer as a *whole* is not the same as n *single* sliders.

Moreover, a production line has an associated product form that indicates the form of the (intermediate) product manufactured in it.

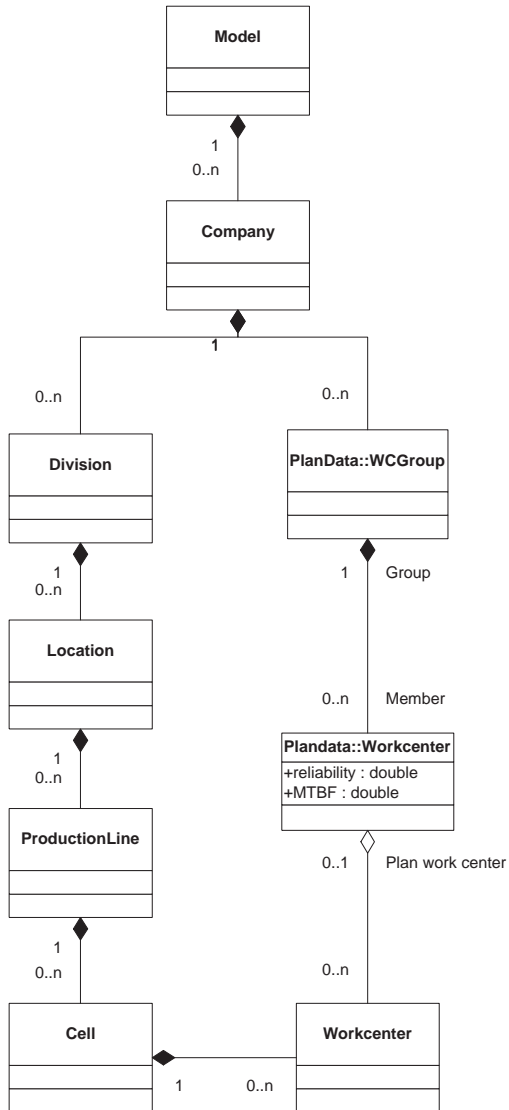


Figure 4.6.: Plan work center

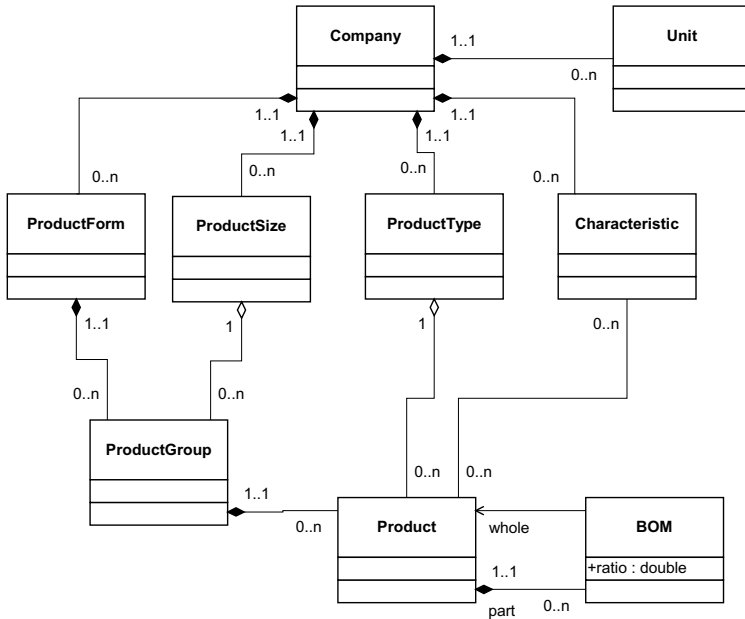


Figure 4.7.: Products

2. **ProductSize**. The product size is used as a dimension associated with products. Each size specifies the conversion ratios of different units, e.g. for products of size **nano** a wafer consists of n sliders, for products of size **pico** a wafer consists of m sliders (see figure 4.8).

Product groups are used to group technically similar products. The benefits of this aggregation are that engineers do not need to specify parameters redundantly in the tool-parameter-sheets and that the simulation model contains less products and can be simulated faster.

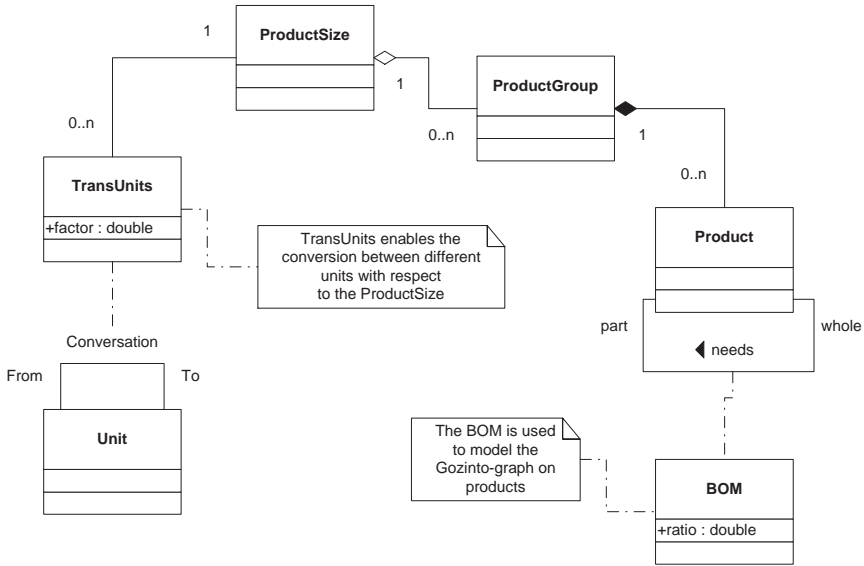


Figure 4.8.: Bill-of-materials

Products are described by their type and some characteristics:

1. **ProductType**. A product type is the primary characteristic of a product. By modeling it as a class of its own, it is assured that each product has exactly one associated type (integrity constraint).
2. **Characteristic**. Whereas product forms, product sizes, and product types have special semantics, the many-to-many relationship between products and their characteristics allows to associate arbitrary information with products. The characteristics can be used in the slice operations of a multi-dimensional analysis, i. e. a categorized presentation of the simulation results.

Managing complex product structures and aggregated products requires conversion between different units. As indicated above two different concepts exist:

1. **BOM.** The bill-of-materials (BOM) specifies the ratio of intermediate products that are used to produce end-products (graph between products, Gozinto graph).
2. **Units.** Units allow the conversion between different aggregation levels of the same product.

In the case of strictly linear product structures these two concepts look alike and can easily be mistaken. Figure 4.8 shows their relationship.

The BOM is defined on products: Each product requires certain other products. Product structures are general graphs (Gozinto graphs) that are modeled by the association **Needs** between successive products. Special cases are convergent and divergent product structures (the Gozinto graph is a in or out tree, respectively). The association class **BOM** provides the factor of how many products are needed to manufacture the successive product.

Conversion between units is defined for different product sizes. For each product size special conversion factors are given and stored in the table **TransUnits**.

4.3.4. Volume Plans

A volume plan is a table that defines the quantities of each product to be manufactured during certain periods (see figure 4.9). The class **Plan** represents the table that assigns a quantity and a plan yield for every **Product**, week, and year. The class **Version** collects administrative information on volume plans. These are a release date, a comment, and a version string. Each **Version** is associated to the production line in that the products of the volume plan are manufactured. The **Unit** associated defines a common unit for all products in a volume plan.

Furthermore, a version is of a special type; for example, one can distinguish between *build* and *ship* plans, depending on whether the plan specifies the real production quantities or the demand that has to be shipped and might be satisfied by finished good inventories.

4.3.5. Routing

The routing among the process steps on different work centers describes two important characteristics of the manufacturing process. These are

1. the process plan, i. e. the order of the process steps,

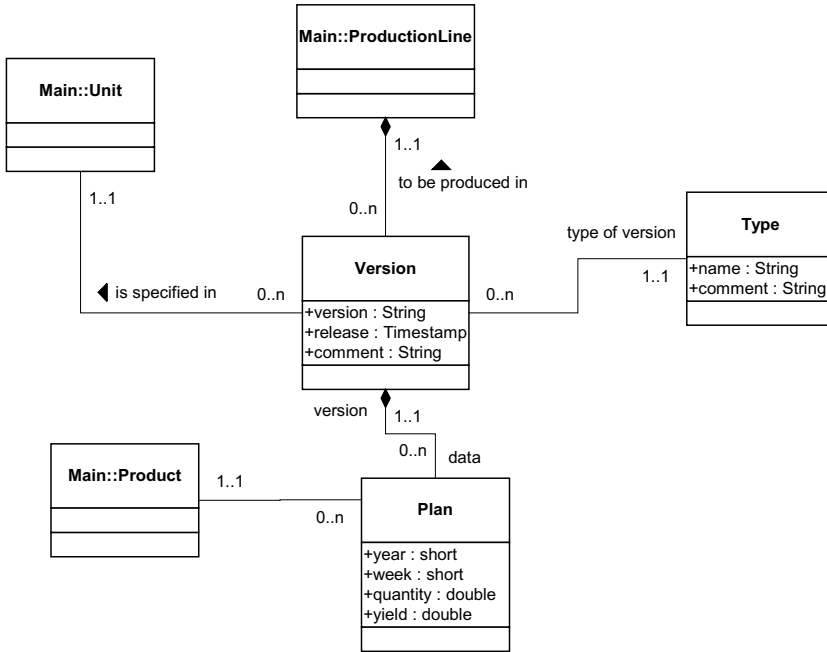


Figure 4.9.: Volume plans

2. and quality aspects (rework, salvage, scrap).

The routing can be thought of as a graph that can be divided into disjunct components. Each component corresponds to a different product group. Each routing, i.e. a step from one process step to the next, is assigned a static probability. The edges describe the process plan whereas the routing probabilities cover the quality aspects. The routing probability can be derived from the first time yield and the throughput yield.

Rework occurs if parts do not fulfill their specifications. These parts are re-inserted into the main process flow at a previous operation. If necessary, the results of the most recent operations are undone.

Salvage, i.e. to get something useful of a difficult situation, means that a production part is actually scrap but its raw materials can be reused.

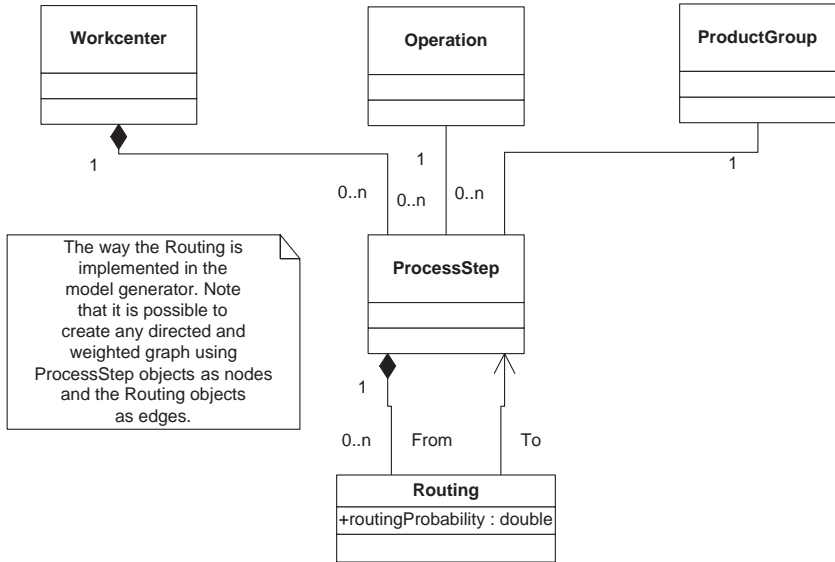


Figure 4.10.: Routing

For example, in the case of a wafer the substrate can be reused. In the simulation model salvage has to be realized as scrap because the wafer has to be processed from the very beginning starting at time 0. It only makes differences to the raw materials planner who does not need to order a new substrate.

Scrap means that parts cannot be recovered. The raw materials and the increase in value of all previous process steps are lost.

The general model of describing the routing as an arbitrary directed graph is shown in figure 4.10. In order to model sensible process plans an additional constraint is required: Process steps can only be connected to process steps of the same product group.

In the underlying database the routing is defined on the basis of the operations neglecting the work centers (see figure 4.3.5). This allows an easy maintenance of the process flow as the work centers can be neglected in the definition of the process flow. A process step is constructed by assigning

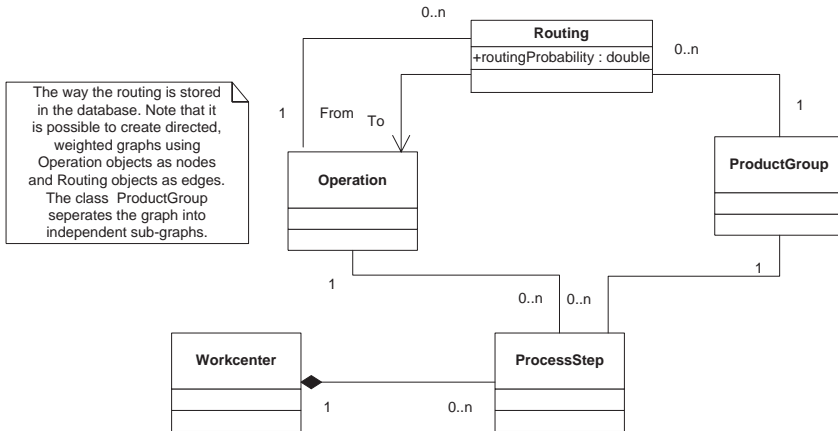


Figure 4.11.: Routing in the database

an operation, a product group, and a work center to it.

The construction of a process plan is not completely defined. If an operation can be performed on more than one work center, the routing probabilities of the routes to the corresponding process steps have to be determined. This problem is described in detail in chapter 10.3.

In order to allow an efficient handling of the routing at the database level and to generate different simulation models, the edges of the routing graph are typed (see figure 4.12):

- *Main flow.* As the order of operations is technologically determined the main flow is normally a linear graph.
- *Rework.* This type indicates the beginning of a rework flow. The routing probability assigned is the first time yield of the operation.
- *Rework flow.* This type characterizes the edges of a rework flow.
- *Back into main flow.* The last edge of a rework flow that leads back into the main flow takes this type.
- *Sector rework.* The operations of a production line are divided into sectors. Rework rates provided by shop-floor-control systems can be

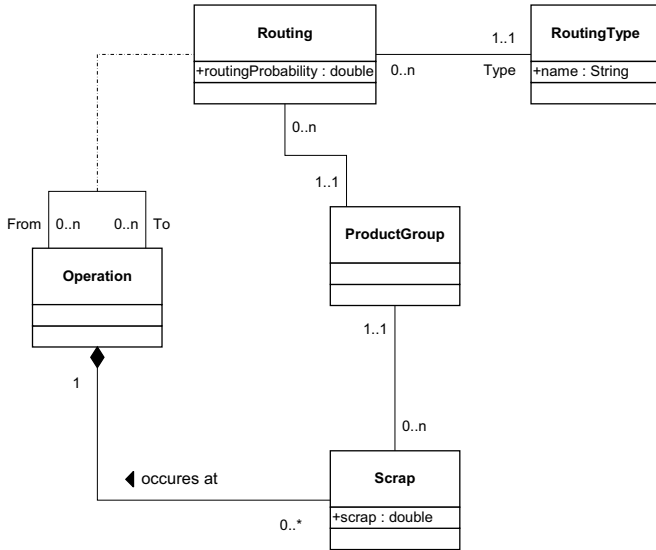


Figure 4.12.: Types of routing

aggregated on different levels. If rework and scrap rates are given at the sector level, it is sometime sufficient to add a single routing from the last operation of a sector to the first. As most sectors are ending in a test operation in order to check the results of the previous process steps, this is an accurate modeling assumption. Taking only sector rework into consideration reduces the number of rework routings and thus enhances the performance of the simulation. Before applying sector rework the manufacturing process has to be analyzed to decide if the assumption is valid.

- *Sector salvage*. This is the rate of scrap parts whose raw materials can be re-used aggregated on the sector level.
- *Salvage*. This routing type indicates salvage on the operation level.

Moreover, figure 4.12 shows the class **Scrap**. Since scrap parts are removed instantly from the production system, scrap is not modeled by an edge

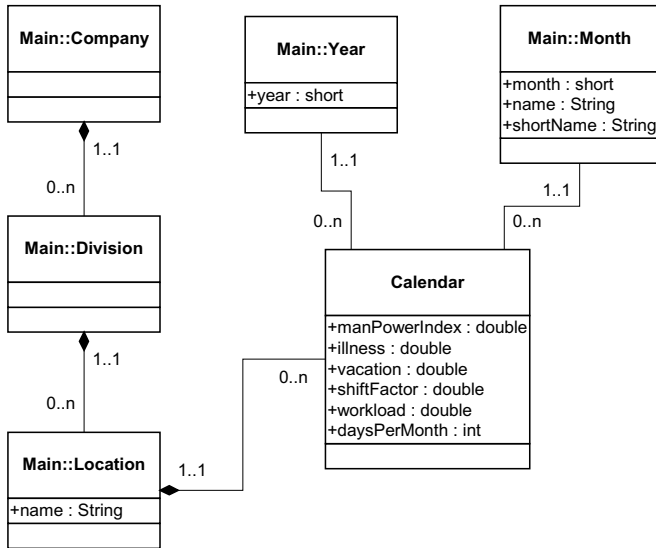


Figure 4.13.: Staffing calendar

but as a value assigned to an operation and a product group. This allows to have different scrap rates at the same operation for different products.

4.3.6. Staffing

To calculate the number of operators needed to manufacture the volume of a build program two additional parameters per work center are needed. The factory or staffing calendar supplies information on vacations, days per month, etc. for each location and month. The static structure of the staffing calendar is presented in figure 4.13. Its attributes and those of the extra work center parameters — `opTimeConst` and `opTimeVar` — are described in table 4.6.

The daily effort, i.e. the amount of time required per day to create all units demanded by the volume plan, at a work center is calculated by the daily going rates (DGR) given in parts and the time needed by the operators

Class	Attribute	Comment
Workcenter	opTimeConst	constant operator time needed per operation
	opTimeVar	ratio of the cycle time the operator is needed
ProcessStep	operatorTime	the time that one operator needs to create one unit
Calendar	manPowerIndex	(or head count availability) amount of time an operator can work per day (in hours)
	shiftFactor	a factor that describes the amount of shifts which are not working. (Example: 3 out of 4 shifts are actually working; one shift is off. The factor is therefore 4/3)
	vacation	percentage of time that operators cannot work because of vacation
	illness	percentage of time operator cannot work because of illness
	daysPerMonth workload	number of days per month number of working-days per specified month

Table 4.6.: Staffing attributes

to perform all operations at the work center:

$$\text{Daily effort} = \text{Operator time} \left[\frac{\text{minutes}}{\text{unit}} \right] \cdot DGR[\text{unit}] \quad (4.2)$$

$$= \text{Operator time} \cdot DGR[\text{minutes}] \quad (4.3)$$

The man power index is the amount of time an operator can work per day. It is given in hours. This yields man power index · 60 minutes per day and operator. The number of operators needed to fulfill the daily effort is the

simple daily head count:

$$\begin{aligned}
 \text{Daily head count}_s &= \frac{\text{Daily effort } [minutes]}{60 \cdot \text{Man power index } \left[\frac{minutes}{Operator} \right]} & (4.4) \\
 &= \text{Operator time} \cdot \frac{DGR}{60 \cdot \text{Man power index}} [Operator]
 \end{aligned}$$

These formulae do not include the times for illness, vacation, etc. These have to be included to get the full daily head count:

$$\begin{aligned}
 \text{Daily head count}_f &= \text{Daily head count}_s \cdot (1 + \text{Vacation}) \cdot \\
 &\quad (1 + \text{Illness}) \cdot \text{Shift factor} \cdot & (4.5)
 \end{aligned}$$

$$\frac{\text{Days per month}}{\text{Workload}} [Operator] \quad (4.6)$$

4.4. The Central Plan Parameter Database

The center of EPOS is a relational database which stores all the information needed for *integrated simulation*. This includes the parameters shown in the previous sections. The collection of plan parameters in a central relational database makes it possible to maintain them company-wide in a non-redundant³, consistent, and efficient way. Many of the parameters can be taken from operational databases like shop-floor-control or ERP systems. Thus, the central plan parameter database serves as a data warehouse (see [HS00]). To load data into the EPOS data warehouse, that data has to be extracted, transformed, cleaned, and united into one consistent model. The integration of data from different sources yields the possibility of integrity checks. This possibility can be extended to create quite powerful validation methods as they are used in the tool-parameter-sheets application (see chapter 5).

When data is maintained in a central plan repository, it cannot only be used as the basis for simulation models but also as a data source for

³In this context redundancy applies only to the data model of the central plan parameter database. As information from other operational databases is used to build this data warehouse the resulting database will be partially redundant with respect to the original databases. Here, redundancy is used as a means of decoupling, enhancing, and integrating external data from other operational databases.

other systems: By the means of standard interfaces like SQL, ODBC, and JDBC all data can be made available for web-reporting, data-mining, etc. (see section 8). As the EPOS data model utilizes multi-dimensional data structures, like the snowflake schema, the data can be used directly in OLAP operations. Practice showed that the use of a central database even without simulation enhanced the transparency of the production process.

4.4.1. Integration of Spreadsheets

Often part of the data needed for planning is stored in a semi-structured way in spreadsheets or ASCII files. This has several disadvantages:

- Normally, information stored in spreadsheets is redundant. This can easily lead to inconsistencies. For example, spreadsheets are often used for volume plans and the name of a product is used to identify its demand⁴. If the name was spelled differently in different versions of the spreadsheet, the demand would account for two different products.
- Spreadsheets contain data that is not in a specified format. Even if some type of format is used, the schema is not stored in a central data dictionary, but it is contained implicitly in the data itself. Changes in the data can therefore lead to changes in the structure thus making unattended algorithmic processing error-prone if not impossible. The format of semi-structured data also changes much more often than the schema in structured data.
- As spreadsheets are normally stored in a file-system, a problem arises when data is changed in different copies of the same spreadsheet. This leads to versioning conflicts.

⁴This will lead to functional dependencies in the data schema and thus violates the second normal form of relational database design.

To circumvent these disadvantages it is the goal to incorporate the data of spreadsheets and ASCII files into the database, as well. This can be done in two ways:

1. Development of a parser which reads the data from a spreadsheet and inserts it into the database
2. Creation of an application which replaces the former need for spreadsheets

Both possibilities are implemented in EPOS. A parser reading volume plans from spreadsheets was developed in order to incorporate data from external sources and store it in the corresponding tables of the database⁵. Another application allowing to manage process flows replaces all former spreadsheet-based equivalents (see section 9.3).

4.4.2. Interfaces to Shop-Floor-Control Systems

Information contained in shop-floor-control (or manufacturing execution) systems can be classified into two groups: information needed to control the manufacturing process on the shop-floor and historical data of manufacturing activities in form of transaction logs or statistical analyses. EPOS contains an interface to the shop-floor-control system MESA (see section 11.1.4) of the IBM wafer line in Mainz. It can be accessed by the EPOS Administrator which is presented in section 9.3. Using this administrator front-end the following data can be loaded from MESA:

- *Operations and sectors.* The definition of operations, their identifiers, and the classification of operations into sectors can be imported for all product groups.
- *Process flows.* A mapping between MESA head types and EPOS product groups allows the import of selected process flows for each product group. The import *wizard* automatically corrects inconsistencies which can arise due to the different structure of the MESA process table.
- *Rework flows and probabilities.* By analyzing the MESA rework statistics, process steps at which rework occurs are identified and rework

⁵Here the problem arises that format of the spreadsheet could change, which requires changes in the parser.

probabilities are estimated from historical data over a time frame which can be freely selected in the import program. During the import process rework flows are automatically added to the main flow.

A simple type of rework which only allows routes within each sector (see section 4.3.5) can be generated from sector summaries: Rework routes are constructed from the last operation to the first operation of a sector. Their probabilities are estimated from the sector's performance statistics.

- *Scrap.* The EPOS scrap table can be filled with estimations made on basis of the MESA history.

To accomplish the import of data from shop-floor-control systems mapping tables for products and work center specific information have to be maintained in order to integrate two different data schemas. On the technical side two relational databases have to be accessed simultaneously. This can be done either by the EPOS Administrator using two JDBC connections or by a database front-end like MS Access using two ODBC connections. The former tool is utilized for standard business processes, the latter can be used for arbitrary imports by administrators.

Chapter 5

Tool-Parameter-Sheets

As not all data can be taken from shop-floor-control systems additional user input is necessary which has to be collected in a well-defined process that guarantees correctness, consistency, and plausibility of the data (see section 2.4.2). Moreover, security aspects and responsibilities play an important role. This chapter describes the features and the technical realization of the tool-parameter-sheets application.

5.1. Overview

Simulation models for production lines require information on the process steps, i. e. which products are to be processed at which work centers, how long does that take (cycle time), etc. This is the most important information for capacity planning as it shows that with varying product mixes the capacity of a production line might change dramatically.

The model of *integrated simulation* describes the need to collect data directly from the responsible engineers and planners. Often companies collect this information manually on sheets of paper and copy it manually into spreadsheets. Obviously, this is rather error-prone and time-consuming. A solution to this problem are the EPOS tool-parameter-sheets. The tool-parameter-sheets application fulfills the requirements of the *integrated simulation* concept. The benefits of the application are:

- Easy roll-out
- Consistency checks
- User-friendly front-end
- Security features like simplified, yet secure access control, electronic signatures, etc.
- Combination of the advantages of relational and document oriented databases
- Special features like work flow support, pictures of work centers, etc.

How these benefits are achieved is shown in the following sections.

5.2. Deployment

Figure 5.1 shows the part of the overall EPOS deployment diagram that represents the tool-parameter-sheets application. Apart from the central EPOS data warehouse (see section 4.4) the Java applet and the Lotus Notes database can be seen. Lotus Notes is a groupware platform that can be used to improve communication, coordination, and collaboration [Tam97, DS97, Kra98]. Information is stored in documents, that allow to be enhanced with typed data fields. Among the available data types are rich text fields that can contain formatted text, pictures etc. The designer of a database specifies how structured information is stored.

As Java applets can be embedded in these documents, as well, the Notes database can be used for the distribution of the applet. Once the end-user opens the document containing the applet, it gets loaded from the Notes server and is run on the client machine. This loading mechanism — shown as the relationship *embedded applet* in figure 5.1 — greatly simplifies the roll-out of the application, as no special client software¹ needs to be installed on client machines.

The relationship between the Java applet and the central plan parameter database is established via JDBC. This is the database middleware provided by Java. The JDBC classes are loaded together with the applet from the Notes server. The plan parameters entered by engineers are written directly

¹apart from the Notes client

into the relational database. Once an engineer has provided parameters, they are recalled from the database when the applet starts. Thus, it is possible to edit the information entered earlier. When the applet connects to the database it checks which work centers are to be loaded and which sections are free for being edited by the current user who is determined by a parameter passed to the Java applet by the Notes environment.

The third relationship shown in figure 5.1, i.e. the ODBC connection between the relational database and the Notes server, establishes reports of the parameters entered into the relational database as Notes documents. The main reason for this are the good publication facilities of the Lotus Notes system. Every five minutes a Notes agent loads the changes from the relational database and creates or updates the corresponding documents in the Notes database. These documents — the tool-parameter-sheets — show all important data of a work center in one document and are managed in different Notes views that allow categorization by locations, operations, products, managers etc. The different views provided by EPOS are presented in section 5.4.

A Notes agent is a program in the Notes environment that can be started

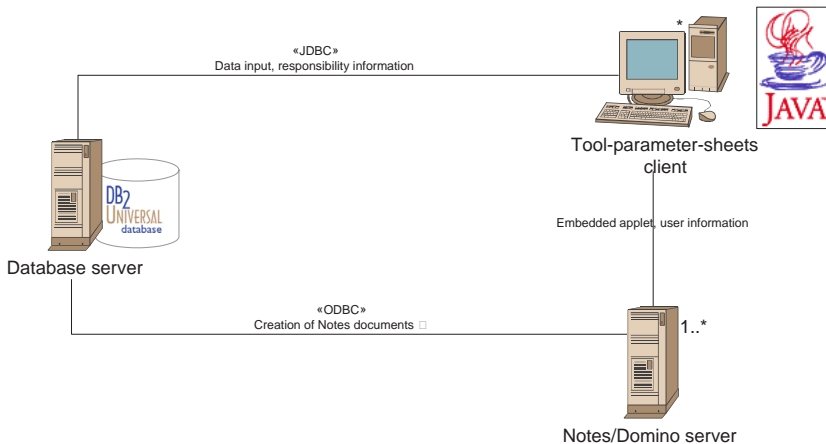


Figure 5.1.: Deployment of the tool-parameter-sheets

by different kinds of triggers, like manual start of execution, a scheduler, or certain events (arrival of mail, creation of documents etc.). Agents can be build in different languages. Simple agents can use Notes macros and formulae, more sophisticated agents can be programmed in Lotus Script or Java. When programming in Lotus Script, the Notes API allows to use ODBC connections. Thus the parameters from the relational database can be extracted and stored in Notes documents.

5.3. Data Input — The Java Applet

After the Java applet has been loaded from the Notes server, the user is presented a list of work centers that he is responsible for, either as main or backup responsibility. Each work center is divided into the four parts *maintenance*, *manufacturing*, *engineering*, and *staffing*. Depending on the specific role that the user is assigned a combination of one or more parts of the work center can be edited. The work centers and responsibilities are assigned to engineers by the logical system administrator (see chapter 9).

The realization of the tool-parameter-sheet application as a Java applet

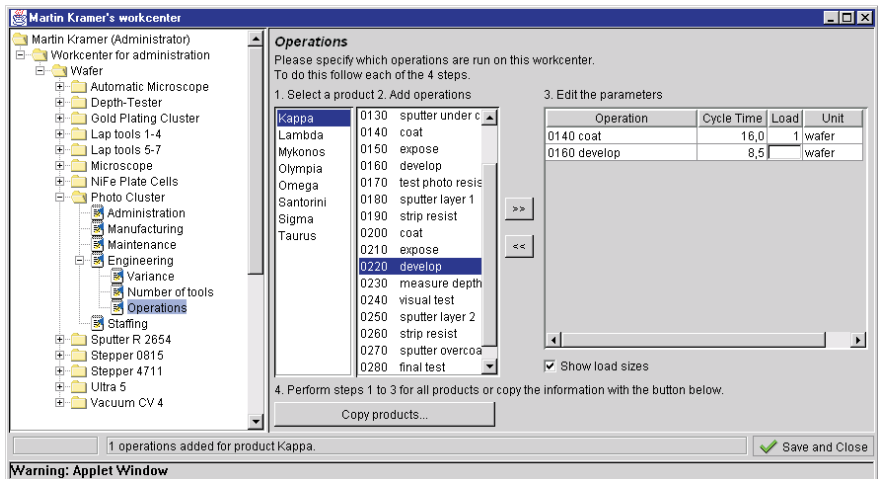


Figure 5.2.: Data input using a Java-applet

offers several advantages:

- *Easy roll-out.* Java applets get loaded from a server. This can be either a web server or the Notes server, for example. No software installation — apart from the client program (web browser or Lotus Notes client) — is necessary. Thus it is very easy to provide the applet to a large number of users.

Especially the database middleware (JDBC in the case of Java) is loaded from the server as well. No installation or configuration has to take place on the client side.

- *Sophisticated access rights.* The Notes environment passes the user's name to the applet. Thus, the applet does not need any further methods for user authentication. Moreover, controlling database access via GUI elements with respect to a special user allows to hide data from the user. He cannot access data stored in the same table he is not authorized to. This would not be possible with tools that allow access to complete database tables like MS Access. This is especially important for the tool-parameter-sheets as normally every user is only assigned few of the work centers. Moreover, only few users usually have access to more than one of the sections (*maintenance, manufacturing, staffing, engineering*). The parameters of different sections are stored in the same work center record, though. In addition, the work center table contains the work centers of different production lines or companies.
- *Typed programming language.* Using a typed programming language with support of exceptions allows to provide the users with a rather stable application.
- *Easy GUI construction.* As the tool-parameter-sheets are used for end-user input, the application should be as user-friendly as possible. Graphical builders for the user interface allow an easy development using standard interface components.

5.3.1. Manufacturing Input

The manufacturing section of the tool-parameter-sheets just consists of the three parameters lunch, breaks, and PFD (personal fatigue and delay):

- *Lunch* [h/day]. This is the amount of time a machine is idle because the operator is off for lunch. If the machine keeps on processing during the operator's break, the value zero has to be entered.
- *Breaks* [h/day]. This is the parameter for break times. As for lunch only those intervals are accounted for in which the machine is idle due to the operator not being available.
- *PPD* [h/day]. This is the time a machine is idle due to personal fatigue and delay. Again, if the machine keeps processing the time should be set to zero.

EPOS does not consider operator availability and parts availability.

5.3.2. Maintenance Input

This section explains the data input for maintenance parameters. All parameters are given per tool, even if the work center contains more than one tool. The number of identical tools for the work center is entered in the engineering section (section 5.3.3)

- *Preventive maintenance* [h/week]. This is the amount of time a tool of the work center is down due to preventive maintenance. The value has to be given in hours per week.
- *Unplanned maintenance*. This parameter characterizes a work center's unforeseen down times due to machine-breakdowns. The process of alternating up and down times is described by the mean time to repair (MTTR) and mean time between failures (MTBF). The values given should be long-time averages, plan data, or assumptions for new tools. Let u_i and $d_i, i = 1 \dots n$, be the length of the up and down time intervals. Then MTBF and MTTR are computed as

$$\text{MTBF} = \frac{\sum_{i=1}^n u_i}{n} \quad \text{and} \quad \text{MTTR} = \frac{\sum_{i=1}^n d_i}{n}.$$

Given these values the average down time per week d^W is calculated by

$$d^W = \frac{168 \left[\frac{\text{h}}{\text{week}} \right]}{(\text{MTBF} + \text{MTTR})[\text{h}]} \cdot \text{MTTR}[\text{h}]$$

- *Planned MTTR* [h], *planned MTBF* [h]. These parameters are used for plan actual comparisons with vendor supplied values.

5.3.3. Engineering Input

The engineering section in the tool-parameter-sheets consists of four panels (see figure 5.2):

1. the main panel in which down times due to set-up, monitoring, engineering, and load size information are specified,
2. the work center variance panel that allows to drag a slider along a scale of variances,
3. a panel that allows to specify the number of tools, and
4. a panel to define the process steps performed at the work center.

The parameters to be specified on the first panel are:

- *Set-up time* [h/week]. The set-up time includes all down times that are necessary for taking a tool into operation or for keeping it working. This includes changing chemical baths, focus check, swapping lights, refilling photo resist, cleaning of equipment, warm-ups, etc.

The following must not be included: changing targets (this is included in the maintenance times) and chilling (included in cycle time).



- *Engineering time* [h/week]. This is the amount of time for product starts, error checking, experiments, and process improvements.
- *Monitoring time* [h/week]. This is the amount of time not available for production due to monitoring.
- *Max and min load size*. These are the maximum and minimum number of parts that can be processed in a batch run of a tool in the work center.
- *Service type*. This parameter specifies whether the tool processes parts sequentially or in parallel, i.e. if the cycle time changes for different load sizes or not. Typical sequential tools are steppers or microscopes, typical parallel tools are ovens. A tool has the service type *parallel*, if the cycle time is not dependent on the load sizes.

- *Operation set-up.* The operation set-up is part of the cycle time and includes pre and/or post processing work that is not dependent on the load size and that is not included in the engineering set-up time. For example, this time can be used for cleaning parts prior to a sputter run, etc.

The parameters *operation set-up*, *service type*, and the load size parameters are used for the determination of the cycle time for the simulation model. The calculation is described in equation (4.1).

The second panel allows to set a slider for the coefficient of variation (section 3.1.2) for the work center. Instead of having to enter a precise decimal the user can select a slider position on a scale from no variance (for tools with no variance in the processing time like sputter machines for which the process time is given in seconds by technological constraints) to manual operations with high variance (inspection operations, for example).

The number-of-tools panel reflects the history of the number of tools in a work center. In order to keep the data input at minimum, only changes are recorded in the table. An entry of the form (y, w, n) means that there have been n tools available since week w in year y .

The last panel is used for the definition of the process steps. It is divided into two selection lists and a table containing the defined process steps. Recall from section 4.3.1 that a process step is identified by a triple of the form (work center, product group, operation). The work center is the currently selected one, the first selection list defines the product group and the second the operation. The table on the right side of the panel shows the operations and their parameters assigned to the currently selected product. The button  adds a new combination of the selected product and the operation to the table. Pressing  removes the currently selected combination. The basic parameters to be specified for a process step are:

- *Cycle time* (or mean process time). This is the time a part is processed at the work center.
- *Actual load size.* In general, the maximum load size is taken to be the default load size unless this parameter is specified.
- *Load size unit.* This is the unit in which the actual load is specified.

Moreover, work center type specific parameters like sample rates, chill times, apply/develop dedications, etc. can be specified in this input panel. End-users are presented fields for entering these parameters in context with the

real-world work centers. In the automatic model generation the parameters entered are needed in auxiliary simulation objects which are created transparently to the end-user.

As process steps at a work center are often quite similar, they can be copied from one product to another. Pressing the button opens a dialog window with two selection lists. The user can select the source product in the first and the destination product(s) in the second list. It is possible to mark several destination products. Clicking the button adds the corresponding operations in the process step table after the action has been confirmed in another dialog window.

5.3.4. Staffing Input

EPOS offers the subsystem *Staffing* for calculating necessary head counts based on the volume plans. This subsystems relies on two input parameters:

- *opTimeConst.* This is the constant time an operator is needed for an operation.
- *opTimeVar.* This is the percentage of cycle time an operator is needed.

These parameters are explained in section 4.3.6.

5.3.5. Security Aspects

Concerning security the tool-parameter-sheets support the following mechanisms:

- *Authentication.* The application assures that a user who wants to connect to the system can be identified uniquely. One possibility of authenticating a user is the use of an account/password combination. In order to authenticate a user, the tool-parameter-sheets use Notes security mechanisms. The Notes environment passes the Notes name of the current user to the Java applet. Thus, password management is left to Notes. Moreover, the applet checks in the EPOS database whether the user connecting to the system has any access rights at all.
- *Authorization.* Authorization involves restricting access of users only to items to which they have been granted access. In order to establish this EPOS maintains certain responsibilities (see section 4.3.2). The

applet loads the work centers based on the defined responsibilities. Only the sections the current user is responsible for can be edited — the others can just be viewed.

5.4. Tool-Parameter-Sheets — The Notes Database

The documents in the Notes database are first of all a means of reporting, i. e. the data entered by the users is presented in a structured way. The main element is a work center. Each work center gets a Notes document of its own. Figure 5.4 shows on the right-hand side the categorized list of work centers. The navigator on the left-hand side shows the available views on work centers. EPOS offers the following views that allow dynamic categorization and sorting:

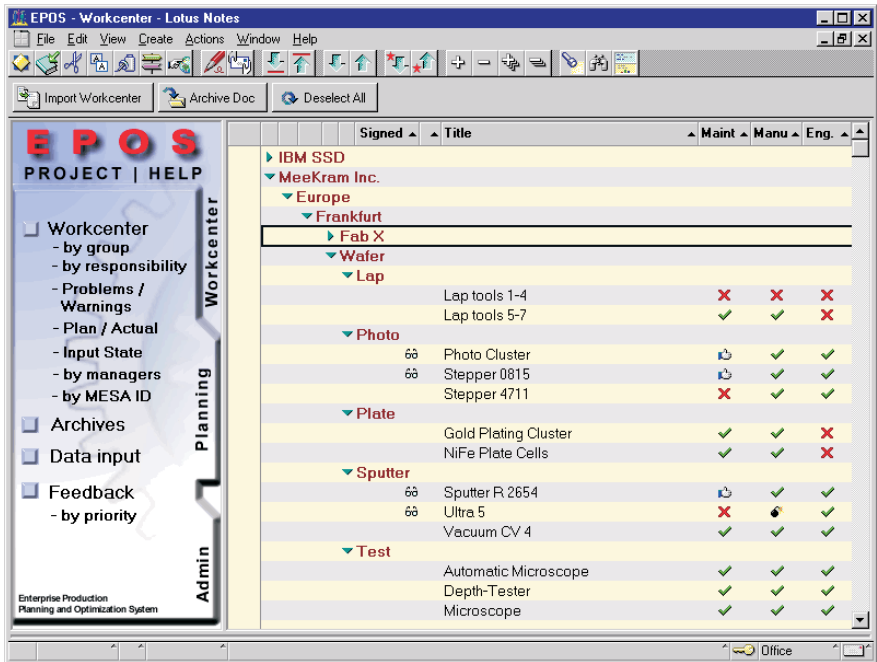


Figure 5.3.: Notes-view of the tool-parameter-sheets

- *Company structure.* This view allows to search work centers by company, division, location, etc. as described in section 4.3.1.
- *Responsibility.* The name of the responsible engineer is shown for main or backup responsibilities. Thus, an engineer can easily find his work centers.
- *Problems and Warnings.* The results of the consistency checks (see section 5.6) are shown in this view. The administrator and the engineers are directly lead towards problems in the data warehouse.
- *Plan/Actual.* This view offers a plan/actual comparison of the work centers' reliability as computed from the data input, PMC specifications, and tool utilization boxes (TUB) (see section 4.3.2).
- *Input state.* This view gives an overview on the number of signed work centers. It provides a means to control the update cycles.
- *Managers.* As a manager has to sign all work centers of his employees, it is helpful to find all those work centers at once. This is achieved by this view.
- *MESA ID.* This view shows the work centers according to their MESA² identifiers which is a more common name for some engineers. Note that there is a one-to-many relationship between work centers and tools (MESA identifiers) as shown in section 4.3.2.
- *Operation.* Operations are performed for certain products on work centers. This view allows to search work centers that perform a special operation.
- *Product.* The same as before, using products as the search index.

Moreover, small icons show the current state of the data input process. The meaning of the icons is given in table 5.1 and the corresponding business process is described in section 5.7. The three columns *Maint.*, *Manu.*, and *Eng.* show the input state information for each section whereas the column *Signed* summarizes the three other flags.

²see section 11.1.4








Icon	Comment
	data input by employee not complete
	data input by employee complete
	signed by manager
	rejected by manager
	some managers signed the work center
	all managers signed the work center
	a manager signed the work center although not all employees declared their input to be complete

Table 5.1.: Icons for work center state

The tool-parameter-sheets in Notes always represent the current state³ as stored in the relational database. In order to keep certain states of the sheets, the Notes database offers the possibility of archiving selected sheets: First of all, the corresponding Notes documents are selected. After the archive button has been pressed the user has to enter a new version string. Then the corresponding documents are copied into the archive that is maintained in the same Notes database as the sheets themselves.

5.5. Use of Different Database Paradigms

Two different database concepts — the relational and the document oriented concept — are used within EPOS. Both concepts have their special advantages and disadvantages which are discussed subsequently.

5.5.1. The Benefits

Relational databases allow to store and retrieve large amounts of structured data efficiently. There exists a proper theory (see [HS00, Vos00b, Ull88], for example) behind the model and nowadays there are several systems commer-

³with a maximum delay of five minutes due to the replication mechanism

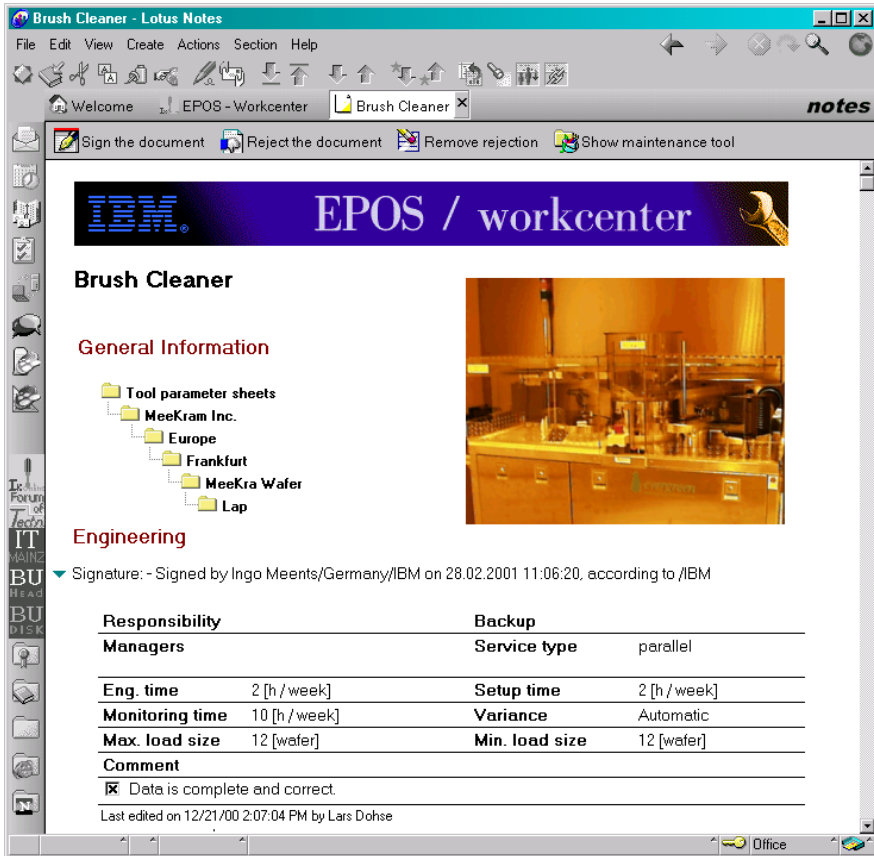


Figure 5.4.: A tool-parameter-sheet as a Notes document

cially available. Thus, all planning parameters are stored in the relational EPOS data warehouse.

Document oriented databases allow to store and retrieve semi-structured data. The workgroup system Lotus Notes is used as the user interface to the data stored in the relational database. Moreover, all additional less structured documents are stored in the Notes-database, like help documents, presentations, etc.

Another important point is that a Notes database can easily be published on the intranet by applying the web-functionality of the Lotus Domino server. Although, if Lotus Notes is already installed, the advantages of using Notes directly should be used. The management of the tool-parameter-sheets as documents in a Notes-view yields many advantages as Notes features like categorizing, sorting, searching, printing, full text search, security mechanisms etc. can be utilized directly (see [Tam97]). The integration into Notes also enhances usability as the sophisticated Notes security mechanisms can be used for the tool-parameter-sheets, as well. This means that the user does not have to go through another authentication mechanism in order to use the system. A single password is sufficient, namely the Notes password that the user has to remember anyway.

As Notes does not have the features required to deal efficiently with large amounts of structured data, the implementation of the data input would hardly be possible directly in Notes. Therefore, the data input is realized by using the additional Java applet.

5.5.2. Replication Between Databases

The deployment diagram in figure 5.1 shows the link between the relational EPOS database the Notes database: There is a one-to-one correspondence of a work center and a Notes document. A Notes agent is responsible for the generation of the work center documents. This is a program that is executed in the Notes environment and can be started manually or by a scheduler. In this case the agent is a Lotus Script program. An ODBC connection to the relational database is opened and a query finds all work centers that have to be updated. The agent realizes an unidirectional replication. In general this could be done by the deletion of all work centers and a successive re-import, but two reasons forbid this:

- *Performance.* It is too time-consuming to re-generate all work center documents regularly, especially as most of the time no changes occur.
- *Information only available in Notes would be lost.* EPOS uses electronic signatures. These are only available in Notes. When a manager signs a document, the signed sections cannot be modified as only the manager with his user-id is able to sign the document. As the signatures should persist until the next update-cycle, the signed work center documents cannot be deleted.

The solution to this problem is a flag in a work center in the relational database that indicates whether the work center has been modified or not. If this is the case, the corresponding work center document is updated, otherwise the work center is skipped. If the work center is updated, electronic signatures for modified parts of that document are lost. This is exactly what is desired as a manager has to check the updated parameters first before he can sign them.

Apart from the work center parameters all information related to work centers is included in the Notes document as well, for example, the process steps, the responsibilities, the company structure, etc. Moreover, all the information that is used for categorization in different views is stored in the Notes document.

5.6. Integrity Checks

As all input parameters are to be processed automatically, the consistency and integrity of the collected data is very important. These are achieved by several means:

- Integrity checks in the database (keys and foreign keys, check and unique constraints)
- A well designed database structure which avoids redundancy and thus ensures integrity
- Management review including electronic signatures for all parameters
- Range checking of parameter values
- Heuristic checks of parameter values and combinations of data entered by different users

These means support the whole process from the data input in the Java applet to the documentation in the archive. The following checks are performed in the Java applet in order to minimize inconsistencies and errors:

- Selection lists avoid free-form user input and establish that product names, operation names, etc. cannot be written differently.
- Boolean input fields force a yes/no choice if applicable.

- Range checking of numerical data assures that parameter values are within their pre-defined ranges.
- Warnings about possibly incorrect parameters specifications are issued.

As not all data is entered with the help of the Java applet it is necessary to establish further consistency checks in the database:

- Range checking for numerical parameters can be realized by check constraints.
- Time intervals can be checked against the assumed units, for example the set-up time that has to be given in hours per week must not exceed 168.
- Combined checks can be performed. For example, the sum of all manufacturing down times (PFD, lunch, and breaks) must not larger that 24, as the times are given in hours per day.
- Logical constraints have to be satisfied, e.g. the minimum load size must not be larger that the maximum load size. Moreover, responsible persons are not allowed to sign their section unless all values have been specified. For the manufacturing section this is established by the check constraint:

$$(\text{READY_BY_MANU} \neq \text{True}) \quad \text{or} \quad (\text{not } ((\text{LUNCH is null}) \text{ or } (\text{PFD is null}) \text{ or } (\text{BREAKS is null})))$$

Whereas the previous checks find data that obviously violates integrity, heuristic checks are used to warn of potential constraint violations. The following warnings are issued if applicable:

- The cycle time of an operation at a work center is negative or equal to zero.
- More than 4 hours of manufacturing times (lunch, PFD and breaks) are entered for each day.
- More than 12 hours of maintenance time are entered for each day.

- More than 6 hours of engineering, monitoring, and set-up are entered for each day.

These warnings are generated by a view in the relational database. The last three checks are not really errors as the given parameters might exceed the given thresholds. But for extreme values, for example, if a responsible person swapped the MTTR and the MTBF values, the warning issued is quite valuable.

5.7. Business Process for Plan Parameter Input

EPOS allows permanent updates of the planning parameters. But sometimes it is necessary to explicitly force all responsible persons to check the plan parameters. This is depended on the industry and on the kind of responsibility. Concerning the semiconductor industry fast technological progress and short product life cycles require more frequent updates by the engineering department in contrast to the manufacturing department as break times etc. are set according to contracts and thus do not change that often. The update process of the plan parameters is a two-stage process: First of all engineers, manufacturers, maintainers update the tool-parameter-sheets. Then the corresponding managers approve the data entered and sign it electronically.

This is realized in two different systems. Whereas the first step is carried out in the Java applet, the second step including the electronic signature is done in Lotus Notes. Figure 5.5 shows the complete business process .

The process starts with the withdrawal of the flags of the three sections *manufacturing*, *maintenance*, and *engineering*. Table 4.2 shows the flags *readyByMaint*, *readyByManu*, and *readyByEng* as attributes of the class **Workcenter**. Moreover, the electronic signatures in the Lotus Notes database are removed. Now the responsible persons are notified that they have to update their parameters. When this is finished, the managers have to sign the data entered. First, the manager of the maintenance department has to sign the parameters, followed by the manager of the engineering department, and finally the manufacturing managers have to accept the parameters. If one of the managers is not content with the parameters for a certain work center, he can reject them. Then the responsible persons have to go through their parameters again until the managers accept them.

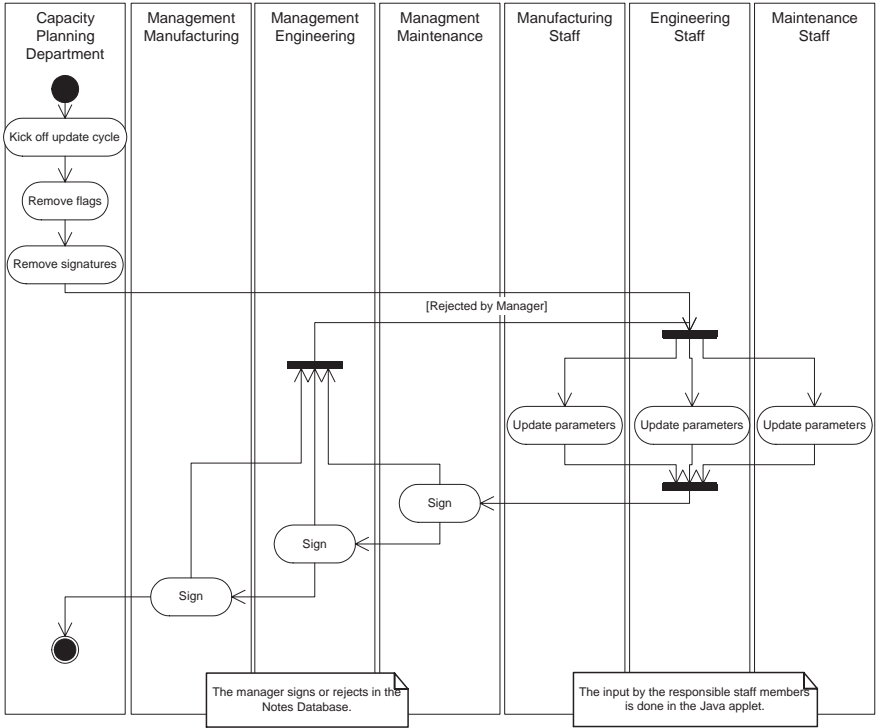


Figure 5.5.: Business process for the distributed parameter input

Chapter 6

The Simulator

The core of any simulation system is the simulator itself. The model of *integrated simulation* imposes some requirements for simulators which might be used in such a system (see section 2.7.4). This chapter presents the core EPOS simulation environment — a client/server system consisting of the simulation server and the EPOS Analyzer, the simulation client. The simulation server has been implemented at the Technical University of Clausthal by Thomas Klein [Kle00a] based on the Analytical Modeling System (AMS) [HS90] and its implementation for the X-Windows environment (XAMS) [HZ97b, HZ97a], the simulation client by Jens Rehaag [Reh00]. In this chapter it is discussed in how far the requirements of *integrated simulation* are fulfilled, different simulation tasks are supported, and how the simulator is integrated into the system.

6.1. Simulation Server

An important requirement stated in section 2.7.4 concerns performance, i.e. to be able to calculate various what/if scenarios in ongoing production planning processes. The time it takes to obtain statistically relevant information from simulation runs has to be extremely short. This is the main reason why the simulation server relies on queueing theory formulae. Advanced mathematical models (see [BT89], [Han98], [Zis99]) enable the

calculation of performance measures for a wide range of scenarios found in today's production. The formulae of the server are able to handle networks of multi-product queues at work centers supporting bulk arrivals and batch services with independent, generally distributed inter-arrival and service times, formally $G^X/G(b,b)/c$ systems. The model allows for machine breakdowns, arbitrary routing graphs including scrap and rework, and the possibility of modeling joint productions. The mathematical background including the network decomposition methodology used is presented in chapter 3.

Another requirement is the possibility of integrating the simulator into the framework of the system for *integrated simulation*. An interface must be provided to create or load models, start the simulation, read back the results, etc. (see section 2.7.4). The EPOS simulation server supports the integration by its CORBA¹ interface. The middleware architecture CORBA is a specification of the Object Management Group (OMG), an organization of nearly 800 member companies which produces and maintains a suite of specifications that support distributed, heterogeneous software development projects. As the OMG only produces and distributes specifications, not software, an implementation of the ORB² needed for the marshaling and de-marshaling process of objects is required. Klein [Kle00a] has chosen the freely available and CORBA compliant implementation MICO (see [Röm01]). Using the public interface of the simulation server specified in the interface definition language (IDL) a software system using any CORBA compliant ORB can access the whole functionality of the server (see appendix C for interface definition). Because of the application interoperability of the CORBA architecture it makes no difference in which programming language the system is programmed or on which platform and operating system it runs.

As the discussion of CORBA is beyond the scope of this work the reader is directed to the OMG [Gro01] or standard literature (see [Röm01], [Ahm98], and [HV99], for example).

6.1.1. Deployment

The deployment diagram of the simulator used in EPOS (figure 6.1) shows four hardware nodes for the EPOS subsystems. It is possible to run these on a single Unix³ machine, but normally they reside on different computers

¹Common Object Request Broker Architecture

²Object Request Broker

³Linux and AIX are supported.

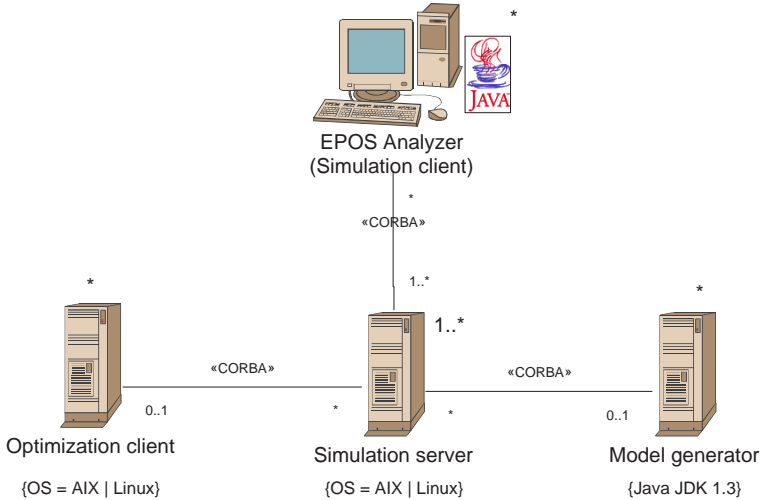


Figure 6.1.: Deployment of the simulator

running different operating systems. The simulator itself consists of the simulation server and the EPOS Analyzer, the simulation client. These two nodes are sufficient for simulation and correspond to a normal desktop simulation system (in fact, server and client perform very well on a standard laptop running the operating system Linux). The optimization client and model generator are included in the diagram to show the integration in the overall system.

The simulation server is implemented in C++ and can be run on a Unix system, it has been tested on Linux and AIX. As only standard components like the standard template library (STL) have been used porting the server to any other Unix system supported by MICO should be possible (see [Röm01]). To use the server's functionality the language-neutral interface definition first has to be compiled into the client's programming language. In the case of Java which is used for the EPOS Administrator, for example, the following command

```
idlj ams.idl
```

creates all Java classes needed for the client from the IDL file `ams.idl` (see

[Sun00b]). To get access to the controller object of the server just the following lines of Java code are needed:

```
[01]    try{
[02]        String args[] = new String[0]; //arguments for the ORB
[03]        ORB = org.omg.CORBA.ORB.init(args, null);
[04]        org.omg.CORBA.Object obj = ORB.string_to_object(ior);
[05]        org.omg.CosNaming.NamingContext namecontext =
[06]            org.omg.CosNaming.NamingContextHelper.narrow(obj);
[07]        org.omg.CosNaming.NameComponent namecomp[] =
[08]            {new org.omg.CosNaming.NameComponent("AMS", "")};
[09]        Ams.Controller controller = Ams.ControllerHelper.narrow(
[10]            namecontext.resolve(namecomp));
[11]    }catch(org.omg.CORBA.COMM_FAILURE e){
[12]        e.printStackTrace();
[13]    }
```

In line nine a controller object which can be used to load or create a simulation model is obtained. To understand the way to get there the following annotations are useful:

- In line three the ORB is initialized using the arguments in the string array `args`. Normally no arguments have to be specified.
- A first reference to a remote object is constructed in line four. A slight inconsistency in Java's and MICO's CORBA implementations makes it necessary to create this first reference via the string representation of the object on the server. The variable `ior` contains a string of 324 characters in which the location of the server object is coded. This so-called IOR (initial object reference) is created when the server is started and does not change (even if the server is rebooted) until the server's network settings are changed.

By using the ORB's `string_to_object` method the general CORBA object `obj` is constructed. This has to be done just once because all other object references are returned directly by the methods of the objects, the controller's `load` method returns the reference to a model object, for example. This leaves the question how the IOR is transferred from the server to the client. In general, this can be done by copy and paste directly into the code of the client or by file transfer. In EPOS all clients can query the main database where IOR strings of available servers are stored.

- In lines five and six the generic CORBA object `obj` is *narrowed* to its proper type `NamingContext`. This can be accomplished by the helper class of the naming context. The object `obj` is now a naming context which is used to obtain the AMS controller in the next step. Note that the IOR does not point to an object of the server, but to the CORBA naming service.
- The following lines, seven and eight, are needed to construct a path to the simulation server which can be resolved by the naming service. Each element of the array is part of the fully specified path. As the server is just registered as "AMS" only one element is needed in the path.
- Lines nine and ten finally retrieve the first object of the simulation server, the controller. It is resolved by the naming service which returns a generic CORBA object. In order to make use of the object it is narrowed to its proper type.

6.1.2. Static Structure

In the last section a way to gain access to the controller of the simulation server from the client side was introduced. To make use of the server its static structure has to be understood. The structure is shown in figure 6.2 which contains the most important attributes and methods, the attributes are further explained in table 6.1. The singleton class `Controller` enables the access to a simulation model. The controller object can be used to load or create `Model` objects. The method `list_Names()` returns the names and further information on all existing models.

A `Model` object represents a simulation model. It contains objects of the work centers and products which are used in the queueing model. Each `Product` object contains a list of operations needed to create one unit of that product. `Operation` objects are assigned to exactly one `Workcenter` object to obtain a valid model. The construction of a process flow is made possible by means of `Route` objects. Each `Operation` object contains a list of routes leading to the successor operations. Each route contains the corresponding routing probability. The successors of an operation have to belong to the same product as the predecessor operation. The sum of all routing probabilities at a single operation has to be less than one. The difference between the sum and one is the scrap factor at that operation.

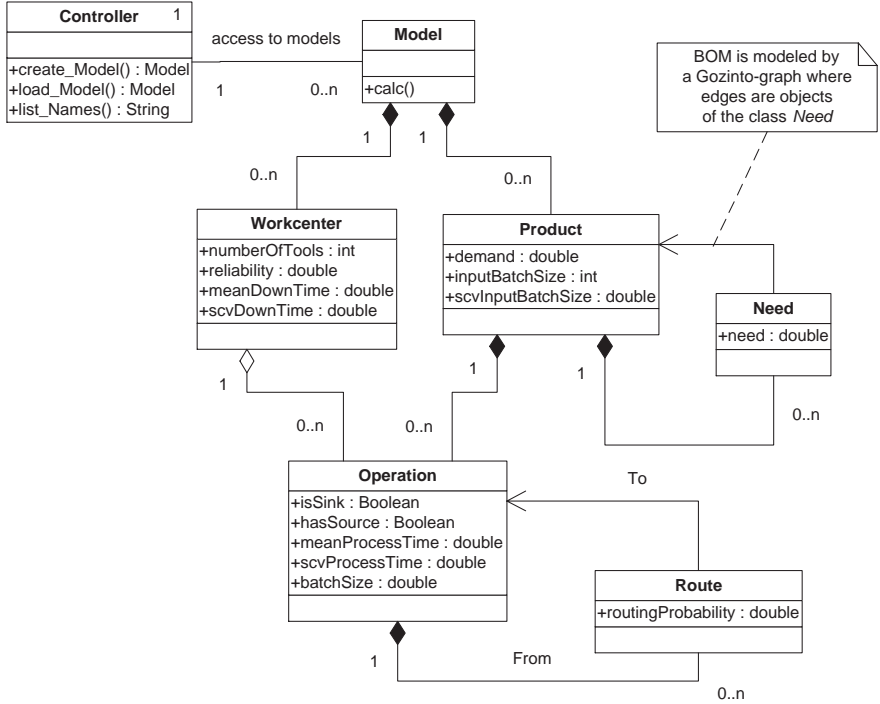


Figure 6.2.: Static structure of the simulation server

This architecture allows to model arbitrary graphs/networks. A product corresponds to a connected subgraph of the complete routing graph.

To model joint productions a product structure can be established by *Need* objects. These objects indicate that for the production of the first product a second product is required. The attribute *need* of type double specifies how many parts are needed. This architecture allows the construction of arbitrary Gozinto-graphs (see [GT00]), i. e. converging, diverging, and general product structures.

To clarify the use of the CORBA interface the following lines of Java code demonstrate how a simple model can be created. It is assumed that the connection to the *Controller* object of the server has been established like it has been shown in the beginning of this section. To create a new

Class	Attribute	Type	Comment
Workcenter	numberOfTools	integer	c_k , the number of machines which are part of the work center
	reliability	double	r_k , the reliability of the work center
	meanDownTime	double	the mean down time $E[MTTR_k]$
	scvDownTime	double	the squared coefficient of variation $C^2[MTTR_k]$
Operation	isSink	boolean	the operation is a sink of the sub-network
	hasSource	boolean	the operation is connected to the source of the product
	meanProcessTime	double	the mean process time $E[S_k]$
	scvProcessTime	double	the squared coefficient of variation of the process time $C^2[S_k]$
	batchSize	integer	the mean batch size of the operation; has to be identical for all operations on the same work center.
Product	demand	double	the primary demand $1/E[I_l]_l$ per time unit of that specific product
	inputBatchSize	integer	the mean batch size at the source of the product's network
	scvInputBatchSize	integer	the squared coefficient of variation of the batch size at the source

Table 6.1.: Attributes of the classes of the simulation server

model the controller is used:

```
Ams.Model model = controller.createModel();
```

The variable `model` now contains a reference to the newly created model object on the server. This model is populated with work center objects:

```
Ams.Workcenter wc1 = model.createWc();
wc.setName("Wc 1");           //name of the work center
```

```
wc.setDesc("the first work center"); //work center description
wc.setTools(2);                      //the number of machines (tools)
wc.setRel(0.8);                      //reliability
wc.setMdt(10);                      //mean down time
wc.setSCVDt(1.5);                   //squared coefficient of the down time
```

The first line is needed to create a work center object in the model, its parameters are set in the following lines. To simplify the initialization the `set` method can be used:

```
Ams.Workcenter wc2 = model.createWc();
wc2.set("Wc 2", "", 1, 0.85, 4, 0.7);
```

A second work center, `wc2`, is created in the first line and all parameters are set in the second line. The order of the parameters is identical to the order of the methods used for the first work center. To place operations on the work centers a product object has to be created first:

```
Ams.Product p1 = model.createPt();
p1.setName("P1");
p1.setDemand(1.4);
```

Each product contains a graph of operations. One of them has to be a sink operation that is created in the next step:

```
Ams.Operation sink = p1.createOp();
p1.sink.setSink(true);
```

The sink operation is a virtual operation which must not be assigned to a work center. The next operation to be created needs to be assigned to the work center that carries it out:

```
Ams.Operation op1 = pt.createOp();
op1.setName("Op 1");
op1.setWc(wc1);           // associate this operation to work center wc1
op1.setMPT(2.0);          // mean process time
op1.setSCVPT(1.5);        // squared coefficient of process time
op1.setBatch(2);          // batch size of this operation
```

A further operation to be placed on the second work center is now created. The next lines show how its parameters can be set all at once using the `set` method:


```
Ams.Operation op2 = pt.createOp();
op2.set("Op 2", "", wc2, false, false, 1.8, 1.5, 1);
    //name, description, work center,
    //is source, is sink,
    //mean process time,
    //scv of process time, batch size
```

Finally, routing objects connecting the operations are constructed by the `createRt` method of the class `Operation`:

```
op1.createRt(op2, 1.0); // op1 -> op2 [1.0]
op2.createRt(sink, 0.8); // op2 -> sink [0.8]
op2.createRt(op1, 0.1); // rework [0.1]
```

The resulting process flow of the model is shown in figure 6.3, a cut from an EPOS Analyzer screen-shot.

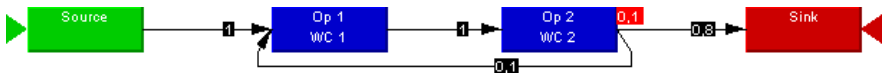


Figure 6.3.: Example of a simple flow

6.1.3. Performance Measures

The simple model created in the last section can be calculated to obtain the model's performance measures. This is simply done by calling the `calc()` method of the class `Model`:

```
model.calc(); // calculate performance measures
```

Even the calculation of the performance measures for large models just takes a few seconds. The following lines of code show how the values can be retrieved from the server:

```
System.out.println("Model: Max. Utilization of = " + model.UtilizationMax());
System.out.println("Wc 1: Work-in-process at work center = " + wc1.Wip());
System.out.println("P 1 : raw process time = " + p1.RawProcessTime());
System.out.println("Op 1: rest lead time of good parts = " +
    op1.RestLeadTimeGood());
```

These lines contain just some examples of performance measures calculated. A complete list is contained in figure 6.4 showing the inheritance relationships among the classes of the simulation server along with the attributes and methods available to retrieve the performance measures. The diagram clarifies the possibility of gaining insight into the model's characteristics at all aggregation levels, for further information the reader should refer to chapter 3 or Klein [Kle00a].

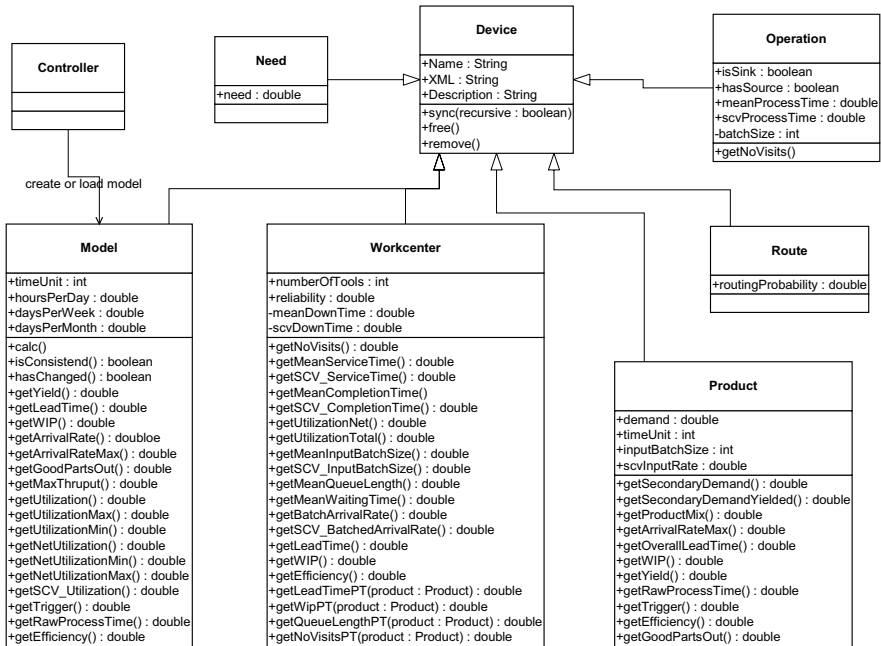


Figure 6.4.: Inheritance, attributes, and performance measures

6.2. EPOS Analyzer

Apart from all automated and integrated processes within *integrated simulation* it is sometimes necessary to carry out certain simulation studies manually. This is the case if questions cannot be answered directly by the

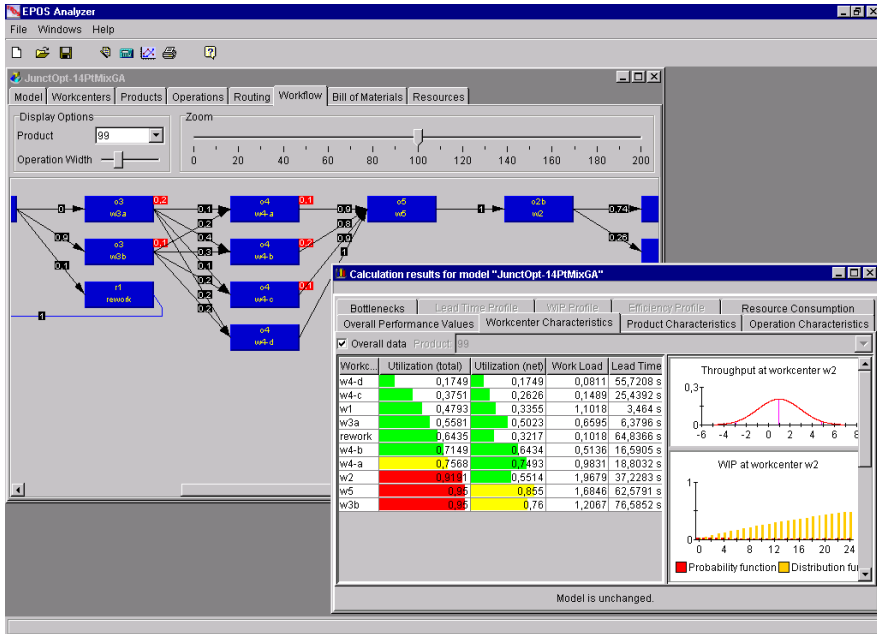


Figure 6.5.: EPOS Analyzer: Work flow and work center results

standard reporting. What/if scenarios demand a tool in which every aspect of the simulation can be controlled. It must be possible to *play around* with a simulation model automatically generated without affecting the contents of the database. This functionality is offered by the EPOS Analyzer, the interactive client for the simulation server. It is implemented by Rehaag [Reh00] in JAVA and accesses the simulation server via its CORBA interface. This makes it possible to deploy the client on any platform that is supported by JDK 1.3 or later. It is even possible to use the client as an applet embodied in a web page.

Using the EPOS Analyzer the planner can build new simulation models from scratch or use simulation models that have been created automatically by the model generator (see chapter 7). While the first approach suffers from the same problems as conventional simulation programs — namely the tedious manual maintenance of simulation models — the second approach

shows how interactive simulation can be integrated into the broader model of *integrated simulation*, the advantages of this integration are quite obvious: The model generator provides up-to-date simulation models at any time. These can be generated from the data warehouse using numerous parameters which influence the kind of model to be created. Then the user can load a model with the EPOS Analyzer and change its parameters in order to obtain a special scenario. Thus, he does not need to maintain the basic models himself and can take advantage of the modeling steps in the model generation and the collaborative maintenance.

The input parameters reflect the class structure of the simulation server. The input window contains a panel for each of the main classes of a simulation model. The parameters of the work centers, products, operations, bill-of-materials, and routings, etc. are maintained in tables. This rather abstract representation allows the experienced user to create and change models very fast.

Simulation models can be exported to and interchanged via XML files. The model export also generates the corresponding XSLT style sheet allowing appropriate viewers like the MS Internet Explorer to show the exported model directly. After clicking on the *Calc* button the performance measures are computed within a few seconds and they are returned from the server for display in a new window. The values are presented on different levels of aggregation, according to the objects of the model. The top ten bottlenecks of the queueing model are presented in a stacked bar chart showing the net and the total utilization of the work centers. All result tables can be sorted which makes it possible to rank the work centers according to lead time or work-in-process.

Moreover, line profiles, i. e. charts showing either work-in-process, lead time, or efficiency⁴ over a specified range of utilization, for example from 10% to 99.5% of the model's capacity can be computed. Profiles enable further insight into the characteristics of a queueing model. Moreover, simulation results can be downloaded into CSV files which can easily be imported into spreadsheet applications like Lotus 123 or MS Excel for further analysis.

⁴the ratio of raw process time to overall lead time

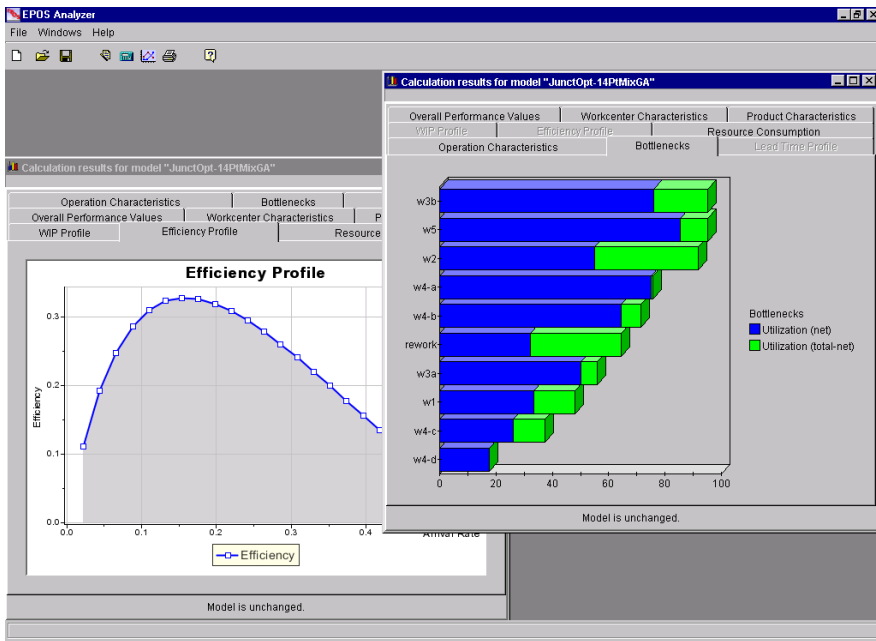


Figure 6.6.: EPOS Analyzer: Efficiency profile and bottleneck chart

Chapter 7

Automatic Model Generation

Automatic model generation refers to the process of generating a model which can be simulated by a simulator. The model of *integrated simulation* requires the possibility of creating simulation models, i. e. models which can be simulated, from information stored in a common data warehouse. The model generation including various aspects is presented in section 2.5. *Automatic simulation* is closely related and identifies the possibility of programmatically controlling the simulation process. This aspect is discussed in section 7.3.

7.1. The Simulation Environment

Using figure 7.1 which shows the deployment of the model generator the general schema of the automatic model generation and simulation can be explained: The administration client, shown on the right side, is used to create so-called simulation requests which contain all information needed to create and simulate models. The administration client — a Java applet which is embedded in a Notes document like the data input of the tool-parameter-sheets application — is explained in section 9.3. Simulation requests which are stored in the central database are the topic of the next section 7.1.2. The model generator periodically scans the database for simulation requests which have to be performed. If a request is found, the model to be simulated is loaded from the database. Then the model generator opens a CORBA

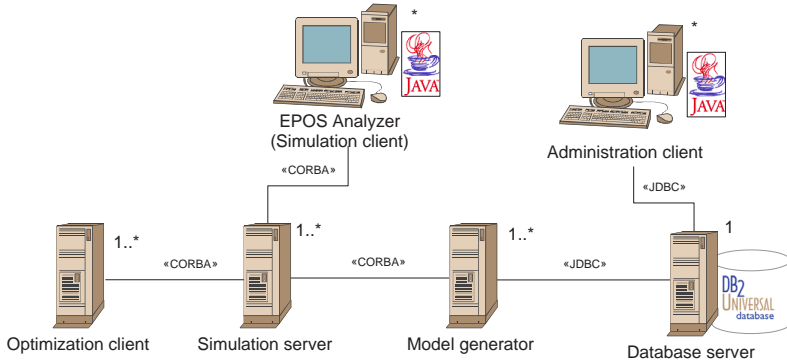


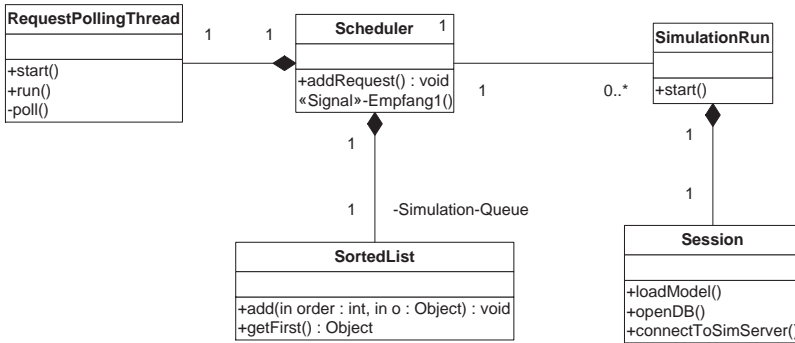
Figure 7.1.: Deployment of the automatic model generation

connection to the simulation server and generates a model which can be simulated using the parameters of the model loaded from the database. This is described in section 7.2.3. After the simulation model has been created on the simulation server it can be simulated for different scenarios defined by the volume plan (see section 7.3). For each simulation run the results need to be collected, aggregated and stored back in the central database which is being described in section 7.18. Moreover, a request might specify to calculate a line profile which again needs several simulation runs (see section 7.3.3).

The simulation models created by the model generator are normally deleted after a successful run. They can reside on the simulation server, though. This yields the possibility for other EPOS modules to use the automatically created models. The UML diagram also shows the EPOS Analyzer, an interactive simulation client which can be used to change parameters and start new simulations. Refer to section 6.2 for a complete description of the client. Another system which can use the generated models is the optimization client explained in section 10.

7.1.1. The Model Generator

The model generator is a Java application which can be run in different environments. It contains a graphical user interface (GUI) to allow end-users to control and monitor the model generation and simulation process. It can also be run without the GUI as a server process, for example on a

Figure 7.2.: The structure of the package **SimSystem**

Linux machine together with the simulation server.

The core of the application is an implementation of the persistent EPOS object model which is initialized from the data warehouse. Via a CORBA connection to the simulation server the model generator is able to control the simulation of generated simulation models.

To start simulation runs automatically the model generator contains a scheduler which static structure is explained by figure 7.2. The scheduler object owns a **RequestPollingThread** object which polls for simulation runs to be started. When this thread recognizes a request to be run the scheduler adds it to its simulation queue. Depending on the scheduling parameters of the request the scheduler creates a new **SimulationRun** object (see figure 7.3). This object is implemented in an independent thread which controls the whole simulation process. It uses its session object to load the model to be simulated from the persistent storage.

7.1.2. Simulation Requests

The scheduler needs to know which models should be created at which point of time. As different simulation models can be created from the information in the real-world model, the transformation process must be parameterized. The scheduler does not only need to know when a specific model should be created, but also how. Thus, the following information needs to be passed to the model generator:

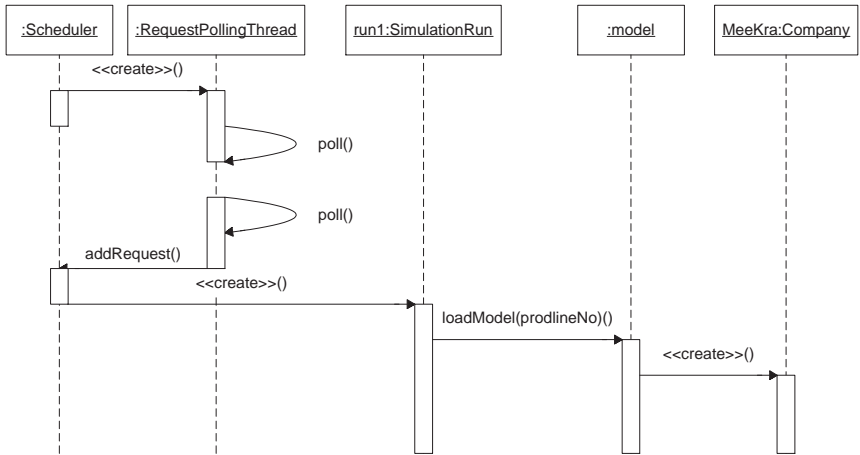


Figure 7.3.: Interaction when a simulation run is started

- Which production line should be simulated?
- When should a model of the production line be simulated?
- Who requested the simulation?
- For which planning horizon should the model be created?
- Which batch size should be used in the simulation model?
- Should the coefficient of variation of the process time specified at the work center be used?
- Which parameters should be used when creating the simulation model?
- Which version of which the volume plan should be used?
- Which additional analysis (e. g. line profiles) should be calculated?
- Which simulation server should be used?

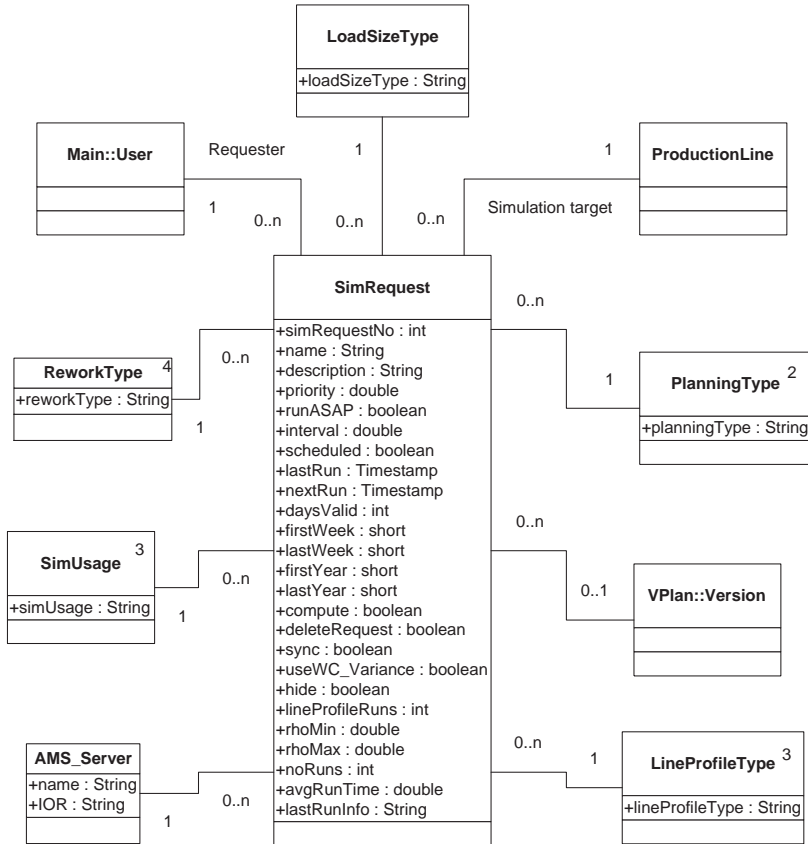


Figure 7.4.: Requests for simulation runs

- Which kind of rework/scrap records are to be used?

EPOS uses so-called *simulation requests* which contain all needed information. An object of the class **SimRequest** is a request to perform a simulation. Figure 7.4 shows the relationship between the class **SimRequest** and the collaborating classes which are mainly used to specify the parameters to be used during the creation of the simulation model.

Each simulation request contains a reference to the person who created the request and thus is the *requester* of a simulation run. To create a **SimRequest** object the user can utilize the EPOS Administrator which is described in section 9.3. The administration applet locates the specific **User** object and associates it with the **SimRequest** object to be created. Depending on the attribute **deleteRequest** the request is deleted after a successful simulation run or resides in the system until it is manually deleted by its creator.

A simulation model is always created for a specific production line. This is reflected in figure 7.4 which contains an association between the classes **SimRequest** and **ProductionLine**. The cardinality of the association also shows that it is possible to maintain more than one simulation request per production line. This is quite reasonable as simulation models created by the model generator can differ because of different parameters of the simulation requests.

The class **LoadSizeType** describes objects for different rules to be used when specifying the process step's batch size during the generation of the simulation model. As the real-world model contains different specifications of batch sizes (see sections 4.3.2) which cannot be specified on some target simulators the model generator needs some kind of rule to determine which load size to use. In the current version a simulation request might specify one of the following rules:

- *Maximum load size.* Use the maximum load size specified at the work center.
- *Minimum load size.* Use the minimum load size specified at the work center.
- *Actual load size.* Use the load size of the process step, if specified. Otherwise use the maximum load size of the work center.
- *Locally optimized batch size.* The batch size is adjusted to the local optimum concerning the queue length at the work center, see section 11.4.1.

An object of the class **PlanningType** is used to specify the planning horizon that the resulting simulation model should be constructed for. Possible choices are *operational* and *tactical*. When *tactical* is chosen some work centers which are grouped in the real-world model are modeled as one work

center. This is done as constraints in an operational scenario might not exist in a tactical scenario (see sections 1.3.1, 4.3.2 and 7.2.9).

A simulation run needs a volume plan which specifies the demand to be used during the simulation (see section 4.3.4). Objects of class `VPlan::Version` describe different versions of volume plans. If a version is specified for a simulation request, the simulation is based on the demand of that specific version. If no version is specified¹, the current version of the production line is used.

Objects of class `LineProfileType` are used to specify the kind of line profile which is to be calculated after the simulation. There are three possible types: *No line profile* obviously means that no line profile should be calculated. *Simple Line Profile* lets the model generator use built-in functionality of the simulation server. If *EPOS Line Profile* is specified the line profile is calculated by consecutively simulating a production line for different degrees of utilization (see section 7.3.3). The advantage of this type of line profile compared to the build-in feature of the simulation server is the access to additional features like multi-process production lines or locally optimized batch sizes.

The association to the class `SimUsage` is mainly used for deployment and testing. Simulation runs for requests specifying *production mode* are used for reporting (see section 8) and are scheduled on the production server. *What/if* requests are also carried out on the production system but will not be used for reporting. Simulation runs for the third type, *Test/Debug*, are not — in contrast to the first two types — carried out on the production system. They are run on test servers.

7.1.3. Simulation Runs

Simulation requests are made persistent in the central plan parameter database. The thread `RequestPollingThread` of the model generator periodically² scans the requests in order to check if a request is scheduled or to be run as soon as possible, has an attribute `nextRun` specifying a time before the current time, and is a request for a production simulation (not a test case).

¹Note the cardinality of the association between `SimRequest` and `VPlan::Version` in figure 7.4.

²Scanning the requests every 60 seconds is normally sufficient.

Attribute	Comment
SimRequestNo	Unique ID of a SimRequest
name	The name of this request
description	A string describing the request
priority	Determines the priority of this simulation request. The higher the priority the earlier a simulation request will be performed.
runASAP	Should this request be run as soon as possible
interval	The interval in which the request should be started
scheduled	Is this simulation request scheduled.
lastRun	When was a simulation for this request run the last time
nextRun	When is the next scheduled simulation for this request
daysValid	How long (in days) will the results of the simulation be valid.
firstWeek, firstYear	The first week of an interval used to limit the horizon of the volume plan
lastWeek, lastYear	The last week of the interval. Leaving one of these values empty results in simulating over the whole period the volume plan.
compute	should the model be simulated (computed) after the model generation
deleteRequest	Should this simulation request be deleted after it is successfully performed
sync	Should the generated model be saved on the simulation server
useWC_Variance	When set to true the coefficient of variation of the process time specified at the work center is used in all operations that are performed on that work center.
hide	Should the simulation run be hidden in the reporting
lineProfileRuns	How many runs should be simulated in order to calculate the line profile
rhoMin	The minimum utilization ϱ_{\min} to be used in a line profile
rhoMax	The maximum utilization ϱ_{\max} to be used in a line profile
noRuns	Number of simulation runs that were already performed for this request
avgRunTime	The average time needed for this simulation run in second
lastRunInfo	Information that was written out at the last simulation run

Table 7.1.: Attributes of class `SimRequest`

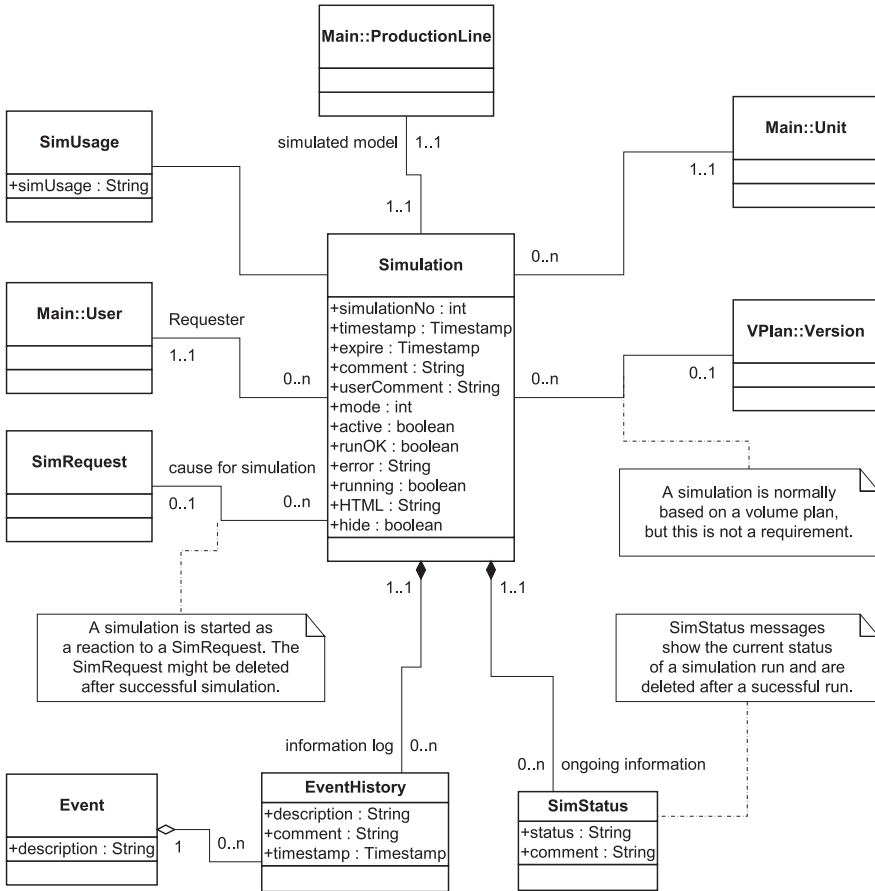


Figure 7.5.: Simulation runs

Simulation requests fulfilling these requirements are sent to the **Scheduler**. Depending on the type of request the scheduler inserts it in a list ordered by the priority of the request or directly starts a new simulation in case the attribute `runASAP` equals true. Each simulation run is carried out in its own thread which is controlled by the class **SimulationRun** (see [OW99]). As the model generator is implemented in a thread safe way all simulation

Attribute	Comment
simulationNo	Unique ID of a simulation run
timestamp	When was this simulation object created
expire	When will the information in of this simulation run expire
comment	A comment created by the simulation server at creation time
userComment	Comments that can be entered by users after the simulation run is performed
mode	An integer that specifies all options of the simulation request
active	Should this simulation run be used for the current reporting
runOK	Was the simulation run successful
running	Is the simulation run currently running
error	Description of the last error that occurred
hide	Should the simulation run be hidden in the reporting
HTML	Description of this simulation run in HTML

Table 7.2.: Attributes of class `Simulation`

requests can be carried out simultaneously. This feature is needed if an user needs a simulation run as soon as possible. This is the case when the volume plan or an important parameter has changed and a fast analysis of this change is needed. Moreover, if a current model is to be analyzed using the EPOS Analyzer, a simulation model should be generated right away. To directly start a new `SimulationRun` the attribute `runASAP` of a `SimRequest` object should be set to `true`. Normally simulation runs are carried out one after another, though. This is done as some large simulation models require a great amount of memory and the simulation result are not always needed directly.

When a `SimulationRun` is started a new `Simulation` object which represents this simulation run is created. Figure 7.5 shows the relationship between the class `Simulation` and its collaborators. Normally a simulation run is initiated by a request. A `SimRequest` object is therefore associated with a `Simulation` object. The simulation request might be deleted before the information about a simulation run should be lost, though. This explains why a `Simulation` object does not mandatorily need an associated `SimRequest` object. This is also the reason why some attributes of the request object are found in the simulation object, as well. These attributes are copied from the request when the simulation object is created.

During the process of model generation and simulation numerous events

like assumptions or error messages need to be logged. The class `Event` is used to specify the type of event that occurred during the simulation run (see section 7.4 for a list of all possible events). Each single event is stored with a time-stamp and a detailed description in an `EventHistory` object. These messages are kept as long as the `Simulation` object, because results of a simulation always need to be evaluated in conjunction with the events that occurred during the generation and simulation. In contrast to this type of information objects of class `SimStatus` are deleted after a simulation run is finished: `SimStatus` messages are just valuable during the actual simulation run, because they contain information about the current status of a simulation run.

7.2. Automatic Model Generation

Automatic model generation in this context means the creation of a model which can be simulated by some simulator. This section starts off by identifying different phases of automatic model generation and simulation: The mechanism used to load the real-world model from the database is presented. Then the creation of a simulation model from the real-world model is described. Different features of the model generator like the way the routing is handled, the ways how special work center types, multi-process production lines, etc. are modeled are explained in detail in the course of this section. The process of simulating the complete planning horizon is covered in the next section 7.3.

7.2.1. Phases of the Automatic Model Generation and Simulation

When a simulation run is created from a simulation request and started by the scheduler different phases of the automatic simulation can be observed:

1. *Set-up.* A session object handling all connections to the database and the simulation server is created and the `Simulation` object is initialized.
2. *Loading the real-world model.* Starting at the top of the hierarchy the real-world model containing the production line which is to be simulated is loaded from the database.

3. *Generating the model.* Using the information in the real-world model a simulation model is derived and generated on the simulation server.
4. *Simulating the Model.* The simulation model is simulated on the simulation server. Parameters that change over the planning horizon are adjusted before the simulation is started.
5. *Saving the results.* After each successful simulation run the results are loaded from the simulation server, transformed, aggregated, and then stored in the EPOS database.
6. *Line Profiles.* Optionally the user can request different line profiles. To calculate a line profile the model is simulated several times for different levels of demand. Aggregated results like the lead time or the work-in-process are stored in the database.
7. *Cleanup.* A report about the simulation run is generated and stored. Then all connections are closed and the memory is freed.

This section deals with the automatic model generation (phases 1–3), phases 4–7 are covered in the section 7.3.

7.2.2. Loading the Real-World Model

Before a simulation model can be automatically generated the objects of the real-world model must be accessible. These objects are stored in the relational database. Thus, all objects must be loaded from the database and connections between the objects need to be restored. Figure 7.6 shows the mechanism which is used to load the objects. A `SimulationRun` object starts the process by creating a `Session` object which is used to open the database connections needed by the `connect()` method. Upon this the `loadModel()` procedure is called which loads the model parameters from the database in order to create the `Model` object. Only the model containing the production line which is to be simulated needs to be created. The `Session` therefore first searches the database for that specific model and only loads the information needed. The mechanism which is used to save time during the loading phase is also utilized for other objects. The `Company`, `Division`, `Location`, and `Process` objects can all be masked, so that only objects which have a direct relationship to the specific production line will be loaded. Other product-related objects like the `Unit` objects do not have such a close relationship and

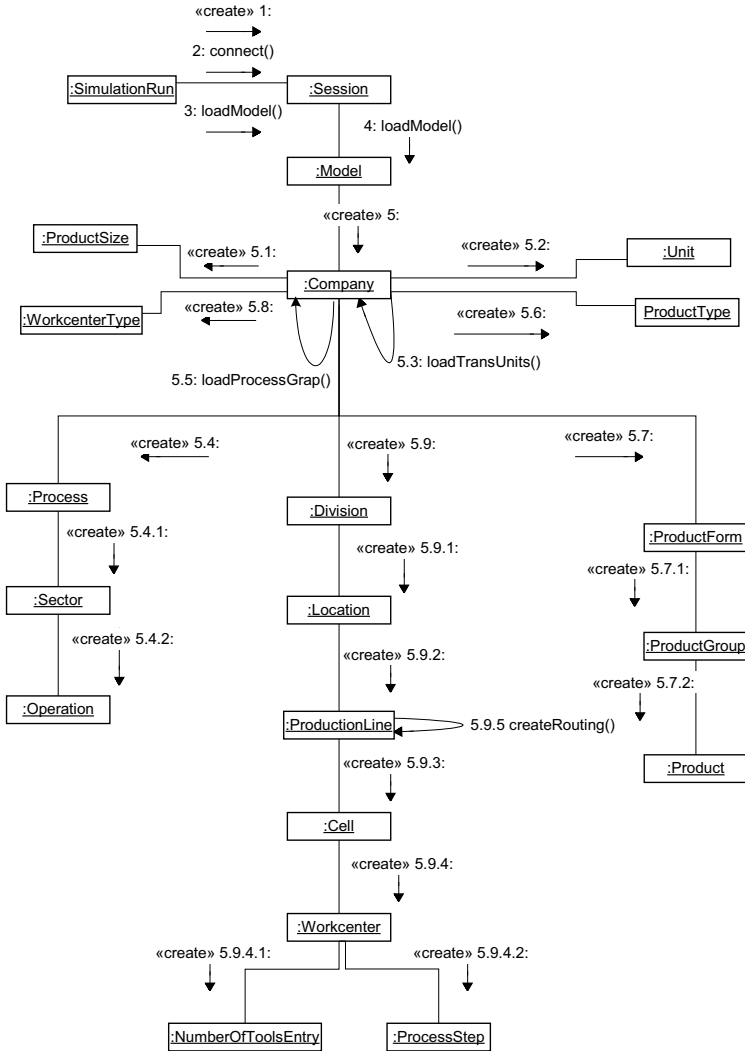


Figure 7.6.: Interaction diagram showing the loading of the real-world model

must therefore all be loaded. The number of **Unit** objects is — compared to the number of **ProcessStep** objects of other companies that might be loaded without this partial loading mechanism — very small, though.

Each object is responsible for creating its direct descendants in the hierarchy of the company structure. The **Model** object creates the **Company** object which in turn creates numerous objects like **ProductSize** or **Unit** objects. In some cases the order in which the objects are created is very important: A company might have several processes which contain the operations needed to produce goods. There can also be references between processes meaning one process is carried out before the other process. These references can only be created, if the referenced processes have already been loaded. This is reflected in figure 7.6 which shows that the processes are loaded in step 5.4 and the graph on the processes in a later step (5.5).

An even more obvious example can be found in the production line. The order in which **ProcessSteps** are performed is described by the means of **Routing** objects which connect one **ProcessStep** (the predecessor) with the following **ProcessStep** object (the successor). To construct the routing all **ProcessStep** objects must be loaded first. This can be done by the production line by loading **Cell** objects which load **Workcenter** objects and so on. When all **ProcessStep** objects are loaded the routing can finally be constructed in step 5.9.5.

The Dummy Work Center

The flow of material through a production line is described by a graph. Nodes are represented by **ProcessStep** objects, edges by **Routing** objects (see section 4.3.5). To create a **Routing** object two **ProcessStep** objects — a predecessor and a successor — are needed. There are several reasons why the routing is not stored in that fashion in the plan parameter database, though (see section 7.2.5). The main difference in the database schema is the absence of work centers in the specification of the routing, where **Operation** objects are used as nodes. Product groups separate the graph into independent sub-graphs. This has two effects when loading the routing from the database:

1. An operation could be carried out on more than one work center.
2. No work center is specified for an operation which is part of the process flow.

The first case can be found in the normal production. Quite often different machines having different parameters can be used for the same operation. Still, routing probabilities which have great effects on the utilization of the work centers have to be specified properly during generation of the simulation model. This problem will be covered in section 7.2.5.

The second case is a result of inconsistent data. If an operation is specified to be in the process flow, it needs some time to be performed³. Moreover, it has to be specified where, i. e. at which work center, the part spends this time, because this means consumption of some resource. This is also true when the specified resource is infinite⁴. This consequently means that a **ProcessStep** object which defines the process time and the work center on which the operation is carried out must exist. If no process step exists for an operation in the process flow, a new **ProcessStep** object is created. A **ProcessStep** object can only be created for a work center, though. An existing work center of the model cannot be used for this purpose as this would falsify the outcome of the simulation. Thus, a special work center — the so-called *dummy work center* — is created. The dummy work center is marked and modeled in a special way during the creation of the simulation model (see section 7.2.3). Its only purpose is to enable the repair of the model (see section 7.4).

Loading Additional Parameters

Some work centers might be of a special type, e. g. *inspections* or *cluster tools*. (See section 4.3.2 for the relationship between **Workcenter** and **Workcenter-Type**.) The usual pattern to implement additional behavior for work centers and process steps in an object-oriented way would simply be to create new classes which are inherited from the classes **Workcenter** and **ProcessStep**, respectively. These new classes would have to load the additionally needed parameters from the database in order to construct themselves. To simplify the implementation of these classes this schema is modified. The additional parameters do not need to be loaded by a class inheriting from class **ProcessStep**; they are loaded in the constructor of **ProcessStep** itself (see section 4.3.2). When a **ProcessStep** object recognizes that the parent

³An operation that does not consume any time or resource cannot add any value to a product. It can therefore not be part of the process flow.

⁴In the simulation this can be modeled using an *infinite server*

work center is a special work center⁵ it creates a hash table that is used to store additional parameters. By calling the method `getParameters()` of class `WorkcenterType` the `ProcessStep` can find out which parameters need to be loaded. These parameters are loaded and then stored in the `ParameterTable`. Classes inheriting from class `ProcessStep` can then query the hash table in order to utilize additional parameters.

Dynamic Loading of the Work Center and Process Step Classes

Another feature to simplify the extension of the model with special types of work centers is the dynamic loading of classes which inherit from the classes `Workcenter` or `ProcessStep`. Before a work center is created (see step 5.9.4 in figure 7.6) its work center type is checked. If the information in the database demands a special work center class to be used, the name of that class is read from the associated `WorkcenterType` object. The same applies to special `ProcessStep` classes. The name of the class which can be obtained from the attributes `WorkcenterClassName` and `ProcessStepClassName`, respectively, is used to dynamically load the class files needed. The following Java code enables the dynamic loading (see [Web98], [Sun00c], [Sun00b], [Sun00a]):

```
String className = getWorkcenterType().getWorkcenterClassName();
Class workcenterClass = Class.forName(className);
Workcenter workcenter = workcenterClass.newInstance();
```

The first line of code is used to get the fully qualified name of the work center class needed from the `WorkcenterType` which is by definition not `null`. Then a `Class` object is created using the `forname()` method. This method uses the system's class loader in order to find and load the class file needed. The next line shows how an instance of the `Class` object can be created with the help of the `newInstance()` method. To make use of the newly created object it must be casted to a class that can be used at design time. In this case a special behavior of work center is needed, the class `Workcenter` is therefore used. The third line of code thus also contains an implicit cast to the super class of the new object⁶.

⁵This is the case when the method `getWorkcenterType()` does not return `null`.

⁶If `Workcenter` is used as the super class, it is obvious that the class of the new object must be (at least indirectly) inherited from `Workcenter`.

7.2.3. Generating the Simulation Model

EPOS uses a simulator based on analytical methods using mathematical results of queueing theory (see section 6.1). This simulator can be accessed via a CORBA interface which makes it possible to seamlessly integrate the simulator. A system for *integrated simulation* needs this tight integration: the client — in this case the model generator — needs full control over the simulation server. It must be able to create models, set the parameters, start the simulation, read the results, etc. If simulation results should be used operationally, an interface via flat files or XML files cannot be used (see section 11). Basically models for every simulator that can be accessed that way could be generated automatically. Moreover, without a tight integration even model files in any kind of format could be created. In the current version of the model generator only the AMS simulation server is supported, though. From now on the description of the generation process therefore refers to this simulator.

After the real-world model has been loaded into memory a connection to the simulation server can be established. A **Controller** object which can be used to create a new simulation model on the server is accessed. Simulation objects corresponding to objects of the real-world model have to be created and filled using the parameters of the real-world model. Hence, the model generation is actually just a *model transformation* as shown in section 2.5.

7.2.4. General Schema of the Model Generation

The smallest entity that a simulation model can be created for is a production line. The next smaller class in the hierarchy, a cell, is used to group work centers. It cannot be used as the basis for a simulation model. This is due to the fact that goods are produced using machines in different cells of a production line. For the simulation to work the parameters of all machines (work centers) in all cells have to be set. Simulation models are therefore created for production lines which contain all work centers needed.

It is possible to simulate the material flow between different production lines of the same company or even different companies which directly leads to questions of supply chain management. Models containing more than one production line could be created, if the real-world model is extended by means describing the flow of information and material between production lines. It has to be evaluated in how far a production line can be replaced by

objects showing the same input/output behavior⁷. However, the simulation of supply chains is not in the scope of this work.

Creating Simulation Objects

Table 7.3 shows the mapping between objects of the real-world model and objects of the simulation server. When creating a simulation model from the real-world model objects are created on the simulation server. Each object on the simulation server (*simulation object*) has a corresponding object in the real-world model. Not every object of the real-world model (*real-world object*) has a corresponding object in the simulation model, though. Firstly, the simulation model does not support all features of the real-world model. The class **Company** of the real-world model, for example, cannot be modeled in the simulation environment. Secondly, even if a feature is supported, simulation objects which do not effect the outcome of the simulation are not created for performance reasons. The mechanism used to skip certain parts of the real-world model is described in the next section.

The simulation is to be run over the time frame of the volume plan. Every week of the plan can be seen as a new scenario. Not only the demand changes, but also the number of tools, the routing, yield and rework parameters. The objects for all scenarios are created before the simulation is started the first time. Then, before each simulation run is started parameters are adjusted according to the specific scenario. That means that a **Workcenter** object is created even though it might not be used during the full time frame of the simulation. In other words: For performance reasons the model is just created once and all changes are modeled by adjusting parameters. A work center which is not used in a specific scenario (number of tools = 0) can be modeled by removing all routes that lead to this work center. As the **Routing** object might be needed in later simulation runs the deletion is modeled by setting the routing probability to a zero.

Top-Down versus Bottom-Up

It would be tempting to utilize the schema used to load the real-world model also for creating the simulation model: One would just use the mapping in table 7.3 in order to create corresponding objects on the simulation server.

⁷In [OB00] artificial neural networks are used to replace the parts of simulation model for performance reasons

EPOS Core	Sim. server	Comment
Model	—	The model is the root of the company structure and contains information which cannot be simulated by the simulation server. Only parts of an EPOS model like production lines can be simulated. The same applies to Company , Division , Location , ProductGroup , Process , Sector , etc.
ProductionLine	Model	A ProductionLine is the smallest entity that contains all objects needed for the simulation.
Workcenter	Workcenter	Workcenter object in EPOS and the simulation server directly correspond to each other.
ProductGroup	Product	In the EPOS model products are grouped to product groups which are used in the simulation, because in many cases the difference between products can be neglected in the context of simulation. To enable simulation of different units in one production line one product for each combination of ProductGroup and Unit used is created on the simulation server (see section 7.2.10).
Operation	—	An EPOS operation does not depend on a product-group in contrast to an operation of the simulation server. An EPOS Operation can be seen as a template which has to be filled with product information to enable the use on the simulation server.
ProcessStep	Operation	An EPOS ProcessStep object is bound to a Workcenter , a ProductGroup and an Operation . This corresponds directly to an Operation of the simulation server
Routing	Routing	EPOS Routing objects directly correspond to the equivalent object of the simulation server.
BOM	Need	An EPOS BOM object is matched to a Need object of the simulation server.

Table 7.3.: Mapping between real-world model and simulation server

Parts of the simulation model can even be generated that easily. First the model, then all products and the bill-of-materials are created. But then a different schema has to be used. The reason why the whole simulation model is not created in this *top-down* fashion (see figure 7.6), but rather *bottom-up*, lies in the model integrity in connection with $1 : n$ relations. The association between the classes **Workcenter** and **ProcessStep** (see figure 4.2), for example, ensures that a **ProcessStep** object always has exactly one parent — a **Workcenter** object. The **Workcenter** object on the other side might have $0, 1, 2, \dots, n$ assigned **ProcessStep** objects. This also includes the case that a work center has no process step assigned. Creating a model in a top-down fashion means that first the **Workcenter** object is created, then all **ProcessStep** objects which are assigned to that work center. Work centers without any assigned process steps correspond to work centers which are not used in the actual production. In a simulation model this means that no part could ever reach this work center. Even if the simulator permits these *empty* work centers, increasing the size of the simulation model with unused objects would be a waste of computing resources.

Constructing the simulation model the bottom-up way avoids this kind of problems. The lowest class in the model hierarchy is **Routing** (see section 4.2). A **Routing** object needs two **ProcessStep** objects which in turn need a **Workcenter**, **Product**, and **Operation** object. These associations (relationships) can be enforced in a relational database by using relational integrity. Basically, the model generation therefore just creates **Routing** objects. Before these objects can actually be created on the simulation server they need to construct the **ProcessStep** objects on either side. These objects create all objects needed themselves and so on. That way process steps which either do not have an incoming or an outgoing routing will not be created on the simulation server. A work center containing only such process steps will not be created, either.

Figure 7.7 shows the process of creating a new **Routing** object. The method `createAMS_Object()` of the source object — a **Routing** object of the real-world model — is called in order to create the corresponding simulation object. In the first step the source object asks its parent, the **ProcessStep** to create its corresponding simulation object. This object needs to be associated to a **Workcenter** object. Thus the process step first calls `createAMS_Object()` of the work center which then creates the simu-

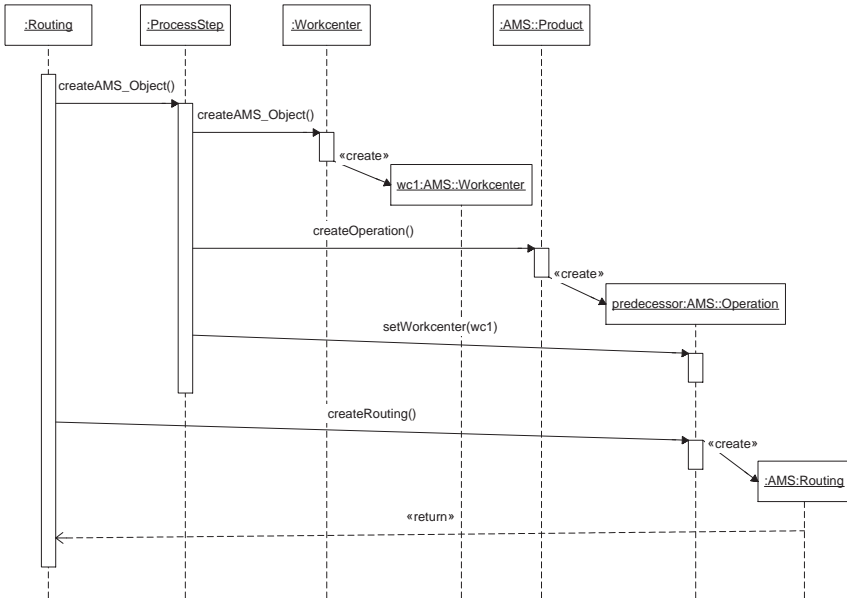


Figure 7.7.: Creation of objects on the simulation server

lation object **wc1**. The process step can now use its product⁸ to create the **Operation** object for the predecessor of the routing. To be fully initialized the **predecessor** operation then has to be associated to the work center just created. Now the **Routing** object can use the **predecessor** to create the corresponding simulation object. This **Routing** object still needs a successor operation which is to be created just like the successor operation.

⁸A **ProcessStep** object has an associated **ProductGroup** object which can be used to get the corresponding product in the simulation model. This object is created in a prior step

7.2.5. Creating the Routing

The flow of material in production lines is modeled by a directed, weighted graph of edges (**Routings** objects) between nodes (**ProcessStep** objects). This graph is divided into disjunct components by product groups. The use of such a graph allows to model rework, scrap, similar operations which are carried out on different work centers (junctions), etc. This simple, yet powerful modeling approach has its disadvantages concerning the maintenance of the graph, though. Problems arise because

1. the amount of rework cannot be set directly i. e. without changing other routing probabilities,
2. the percentage of scrap is only defined implicitly,
3. the distribution of work load between two or more work centers which can perform the same operation is hard to specify.

To bypass these problems the real-world model is extended by routing types and scrap records (see section 4.3.5). Moreover, during the creation of the simulation model further adjustments are needed.

To overcome the first problem it should be possible in the real-world model to set the amount of rework without having to change any other routing probability (see section 4.3.5). Normally, this would require two steps: The routing probability of the main flow would have to be reduced by the rework probability, then this probability would have to be set at the rework edge. Using a rework routing type⁹ it is possible to simply set the rework probability without having to change the probability of the main flow routing. Subtracting the rework probability from the main flow routing is done when the simulation model is created.

The second problem can be tackled by introducing scrap records. These define the amount of scrap for an operation and product group¹⁰. This allows the user to explicitly enter the scrap factor for an operation. The semantics are to subtract the amount of scrap from the main flow routing. Once again, this is done during the model generation in order to create a consistent simulation model.

⁹A routing whose type is set to *rework*

¹⁰For maintenance reasons the work center is not specified, just like it is done for the routing itself.

The third problem occurs when an operation for one product group can be performed on two or more different work centers, e.g. in the example the operation 0815 apply photo resist can be carried out on *photo cluster 1* and *photo cluster 2* for product group *Omega*. In many cases the machine operator can freely choose the machine on which the operation is carried out. The routing probabilities needed could be found by using a statistical analysis of the past. This is time-consuming and not always possible. This distribution is probably not optimal, either. An optimal distribution of work between work centers is influenced by the time-dependent capacity of the work center determined by product mix, batch size, arrival rate, rework, etc. Moreover, different choices of routing probabilities might not be optimal with respect to different optimization goals like lead time, work-in-process, yield, etc. Optimal settings in non-trivial cases can therefore only be found by optimization which is discussed in section 10.3 where a formal model is derived and solved. Due to time constraints an optimization run cannot always be carried out for complex production lines, though. Still, the user should not be urged to maintain parameters that are hard to specify or are even subject for optimization. Thus, some kind of heuristic is needed that can be used to specify the routing probabilities without having to go through a full optimization run is. The heuristic of the model generator uses so-called local capacities in order to determine the distribution between work centers. This is described in detail in the following section.

Comment: *It has to be noted that all routing features described work orthogonal and can be combined. The user could also decide to use non of the features provided and create a process flow using neither rework routings, scrap records, nor automatically specified routing probabilities.*

If it is desirable to specify the distribution of work between two or more work centers, it can easily be done. One would have to split a single operation into one operation per work center. Then it can be specified in the process flow how many parts are sent to each operation or work center, respectively. Moreover, rework operations can also be carried out on more than one work center. Probabilities for rework routings are also set on the basis of the static capacity of the work centers which carry out the rework operations.

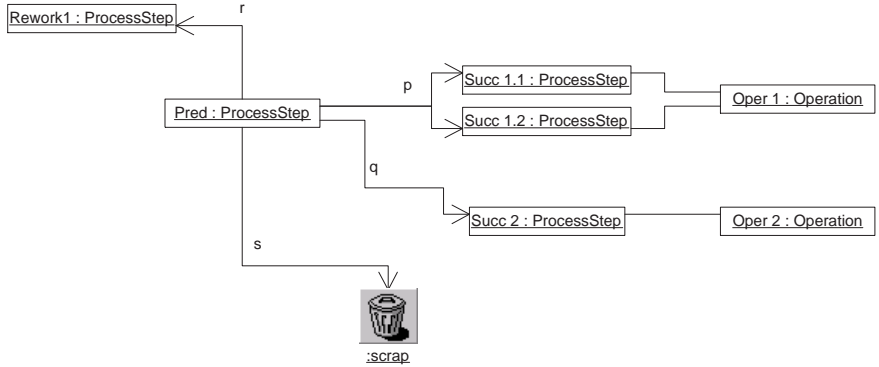


Figure 7.8.: Example of different types of routings between process steps

Calculating Routing Probabilities in the Simulation Model

Routing probabilities of the real-world model cannot be used directly in the simulation model. If rework routings and scrap records are used, the sum of routing probabilities can be greater than 1. The routing probabilities of the simulation model thus have to be calculated from the values specified in the real-world model. Before the general calculation schema is discussed an example is presented: Figure 7.8 shows a sample part of a process flow for one product group¹¹.

Different types of routings and a scrap record are assigned to process step *Pred*. The percentage of scrap is denoted by *s*, the amount of rework which leads to process step *rework 1* is given by *r*. Process steps *Succ 1.1* and *Succ 1.2* share the same operation *Oper 1* but are carried out on different work centers (which are not shown in the diagram). The routing probability *p* specified for the routing to operation *Oper 1* is duplicated during the loading process. Process step *Succ 2*, defined for another operation, *Oper 2*, is defined with another routing probability, *q*. The goal during the model generation is to calculate routing probabilities \tilde{p}_1 , \tilde{p}_2 , \tilde{q} , \tilde{r} for the corresponding routing objects on the simulation server. These probabilities have to satisfy the following conditions which can be derived from the first two problems

¹¹The navigation between objects in this UML diagram also reflects the flow of materials.

stated in the last section:

$$\tilde{r} = r \quad (7.1)$$

$$1 - (\tilde{p}_1 + \tilde{p}_2 + \tilde{q} + \tilde{r}) = s \quad (7.2)$$

The first equation states that the resulting rework probability has to be same as the one set in the rework routing. This addresses the first problem: The rework probability can be set directly in a single place by specifying r . The second equation demands the resulting scrap to be s , just like it has been explicitly set in the scrap record. By specifying the scrap rate in the scrap record, $p + q = 1$ (meaning no scrap is hidden in the main flow routing) can be assumed the following.

After scrap and rework have been specified a routing probability of $1 - (r + s)$ is left for the main flow routing which has to be split between operations *Oper 1* and *Oper 2*. This leads to

$$\tilde{q} = q(1 - (r + s)) \quad (7.3)$$

$$\tilde{p}_1 = x_1 p(1 - (r + s)) \quad (7.4)$$

$$\tilde{p}_2 = x_2 p(1 - (r + s)) \quad (7.5)$$

$$x_1 + x_2 = 1 \quad (7.6)$$

Using $p + q = 1$ the probability \tilde{q} is calculated using the q -part of what is left for the main flow, $q(1 - (r + s))$. Operation *Oper 1* is carried out on two work centers. The other part, $p(1 - (r + s))$, therefore has to be split again using two weights, x_1 and x_2 .

It can easily be shown that by using this definition the routing probabilities and the defined scrap add up to 1:

$$\begin{aligned} \tilde{p}_1 + \tilde{p}_2 + \tilde{q} + r + s &= \\ x_1(p - p(r + s)) + x_2(p - p(r + s)) + q - q(r + s) + r + s &= \\ (x_1 + x_2)(p - p(r + s)) + q - q(r + s) + r + s &= \\ p - p(r + s) + q - q(r + s) + r + s &= \\ (p + q) - (p + q)(r + s) + r + s &= \\ 1 - (r + s) + r + s &= 1 \end{aligned}$$

Still, the weights x_1 and x_2 have to be defined. This is done on the basis

of a so-called *local capacity* function¹² $C_l(t, w, p)$ defined for work center w and product group p at the point of time t . This approach was chosen, because the calculation of weights has to be done fast. The resulting routing probability should also be independent of the product mix. Otherwise each junction would have to be re-calculated for every week the model which is to be simulated. The local capacity of product type u_l at work center w_k and time t is defined by

$$C_{local}(t, k, l) = \frac{b_k r_k c_{k,t}}{\sum_{a_m \in \mathcal{A}_{l,k}} E[S_m] sample_m} \quad (7.7)$$

where

$\mathcal{A}_{l,k}$	set of process steps of product group u_l at work center w_k
b_k	batch size at work center w_k
$c_{t,k}$	number of tools available at work center w_k and time t
$E[S_m]$	mean service time of process step a_m
r_k	reliability of work center w_k
$sample_m$	sample rate of process step a_m .

The local capacity C_{local} is a theoretical upper limit of the number of parts of a single product group that can be produced per time unit on a work center without taking rework, scrap and product mix into consideration. By routing parts to process steps depending on the local capacity of the work center that each process steps is located on the weights x_1 and x_2 can now be defined by

$$x_1 = \frac{C_{local}(t, w_1, p)}{C_{local}(t, w_1, p) + C_{local}(t, w_2, p)} \quad (7.8)$$

$$x_2 = \frac{C_{local}(t, w_2, p)}{C_{local}(t, w_1, p) + C_{local}(t, w_2, p)}. \quad (7.9)$$

Work centers w_1, w_2 denote the objects to which the process steps *Succ 1.1* and *Succ 1.2* are associated, respectively.

¹²The term *local* was chosen, because no information of the collaborating objects like other work centers, process flows, but just local parameters of the work center itself are used in the calculation.

The General Case

For the general case the following formula is used to calculate the transition probability of routing r at time t :

$$p(t, r) = x_{\text{succ}(r)} p_r - \begin{cases} 0 & \text{type}(r) \neq 0 \\ x_{\text{succ}(r)} p_r (\text{rework}(\text{pred}(m)) + \text{scrap}(\text{pred}(r))) & \text{type}(r) = 0 \end{cases} \quad (7.10)$$

with

$w(m)$	work center on which process step a_m is located
$\text{pred}(r)$	the predecessor of routing r
p_r	transition probability of routing r
$\text{rework}(m)$	rework at process step m
$\text{scrap}(m)$	scrap at process step m
$\text{type}(r)$	the routing type of routing r , i. e. if r is a routing in the main flow, $\text{type}(r)=0$.

The weight x_m of process step a_m is calculated by

$$x_m(t) = \frac{C_{\text{local}}(t, w(m), p(m))}{\sum_{a_n \in \mathcal{A}_m} C_{\text{local}}(t, w(n), p(n))} \quad (7.11)$$

where

$w(m)$	Work center on which the process step a_m is located
\mathcal{A}_m	Set of process steps which are successors of a_m .

Finding sinks and sources

The real-world model does not specify the beginnings and ends of the process flows — sink and source operations, respectively. This information is redundant being implicitly stored in the process flow. The simulation server demands sinks and sources to be set explicitly, though (see [Kle00a]).

To find sinks and sources the numbers of routing objects leading into and out of process step objects are analyzed. Finding sinks is quite simple:

Each process step object contains a list with routing objects leading to the following process step. If this list does not contain any main flow routing¹³, the process step does not have any successors and is therefore a sink.

The routing is modeled unidirectional which makes it more complicated to find source process steps as the objects do not contain any information about the number of routing objects pointing to them. To find sources this number has to be calculated which is simplified by the way the simulation model is created: Figure 7.7 shows how all process steps needed, work centers, etc. are created by using a loop over all routing objects. For all main flow routings a counter of the successor process steps is increased. After the loop a source operation can be identified by searching for process steps which have a counter equal to zero, i.e. no main flow routing is pointing to these process steps. These objects are then set as sources for the process flow. It is important to differentiate other routing types from the main flow routing when counting predecessors. The salvage routing type, for example, is used for routings which lead from any arbitrary process step to the first process step in the flow (see section 4.3.5). If salvage routings were considered when counting predecessors, source operations could not be identified.

7.2.6. Modeling Transportation

A main advantage of the automatic model generation is the possibility of modeling transportation between process steps easily. A **Routing** object connects two **ProcessStep** objects specifying that parts are moved from one process step (predecessor) to another process step (successor) at a specified probability. In the simulation model such movements are carried out without the consumption of any resource or time. Depending on the situation the time or resource used during the transportation between process steps can be neglected in the model. It can be important, though, to model these movements when one of the following conditions is true:

- Transportation adds a significant amount of time to the total lead time.
- Transportation requires resources and could even become a bottleneck in the production line.
- A large batch size is used for transportation.

¹³A **routing** object with routing type 0 specifying this routing to be in the main flow

When incorporating transportation in the simulation model two observations are helpful: Firstly, parts are moved between two the process steps which are connected by a routing. Secondly, movements can be modeled by process steps. A process step needs some work center that it is assigned to and some mean cycle time to be completed. The same is true for transportation: Parts need to be moved from one place to another by some means — the transportation work center. Transportation itself requires some amount of time — the process time of a transportation process step. If transportation is significant in few places only, it can be modeled in exactly that way by manually creating a work center and the needed process steps. If it is to be evaluated at each routing of the production line, automatic model generation can perform this tedious modeling task.

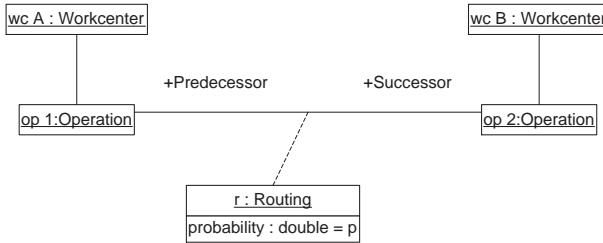


Figure 7.9.: Routing without modeling transportation

Figure 7.9 shows how the **Routing** object r is created on the simulation server. It connects the operations op_1 and op_2 with the routing probability p . These process steps are carried out on the work centers wc_A and wc_B , respectively. To model the movement from op_1 on wc_A to op_2 on wc_B a new process step is created and placed on a new work center (see figure 7.10). This work center models the transportation facilities used in the production line which could be human beings, material handling systems, automatic guided vehicles (AGV), etc. The number of tools t and the reliability rel are set according to the transportation system used. The mean process time pt of the transportation process step can be calculated in one following ways:

- Let d be the distance between the two work centers wc_A and wc_B calculated using some metric — the Manhattan distance for example¹⁴.

¹⁴Right now this is the only method implemented in the model generator.

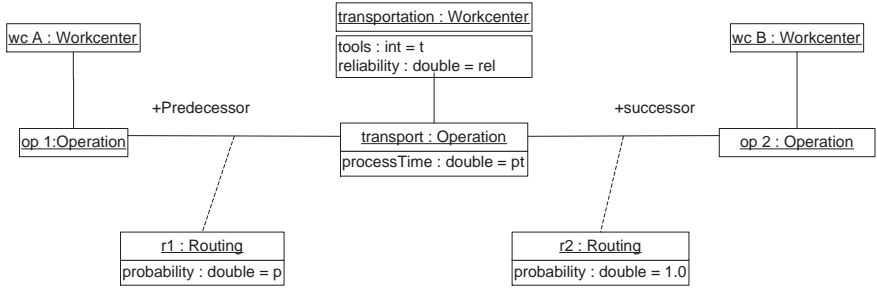


Figure 7.10.: Routing with a transportation operation at the place of a routing

Using the mean velocity v of the transportation facility the process time can be set to $d \cdot v$.

- An extra time could be added when work centers are located in different cells.
- A matrix $(D^w)_{i,j}$ containing the distances between work center i and j could be used to calculate the transportation time.
- Instead of defining a distance between each work center a matrix $(D^c)_{i,j}$ containing distances between cell i and cell j could be used to simplify the maintenance.

Figure 7.10 also shows the second routing r_2 which is needed to connect the transportation process step to the successor process step op_2 . The routing probability is set to 1.0 as the probability defined for the original routing p is already applied at routing r_1 ¹⁵.

Another important parameter to be specified is the batch size of the transportation process steps. It is quite normal in production systems that different batch sizes for production and transportation systems are. Just like all parameters of the transportation work center, the batch size used for all transportation process steps is taken from the simulation request.

¹⁵Note that transportation of scrap parts is not modeled that way, as scrap parts stay at the work center where they were scrapped.

Automatic modeling of transportation increases the complexity of the simulation model by a single work center and one additional operation and routing per **Routing** object of the original model. Results of the transportation work center cannot be stored directly in the central database as the transportation work center only exists during the simulation. Thus, results are documented by creating a general event containing all results of the transportation work center and process steps (see section 7.4.4).

7.2.7. Special Work Center Types

Not all work centers of the real-world model can be modeled by a single work center in the simulation model, for example, a machine that performs various operations in a sequence with the help of a material handling system. Modeling this machine by just one work center in the simulation model would not lead to correct results. This is due to the way the mean process time is specified:

- If the time that a batch needs to perform the whole operation is specified, the utilization of the work center cannot be calculated correctly when the machine processes batches by pipelining. Here the capacity of the work center is determined by the time that is needed for the longest step in the sequence. This is the trigger or bottleneck of the machine.
- If the mean process time is taken to be the longest step in the sequence, the capacity is modeled correctly. However, the time spent for other sub-operations cannot be included in the simulation model, though.

Using an object-oriented discrete event simulator work center objects that model the desired behavior directly can be utilized in the simulation model. Simulators based on queueing theory do not yet provide special types of work centers. Most of the functionality needed can be modeled by the use of sub-networks, though. In the example above the machine could be simulated using two work center objects in the simulation model, one for the trigger (the operation having the longest step in the sequence) and another one for the remaining lead time. One important advantage of the automatic model generation is the possibility of transparently creating and managing sub-networks in the simulation model. The transparency in this process is quite important: Persons entering parameters into the real-world

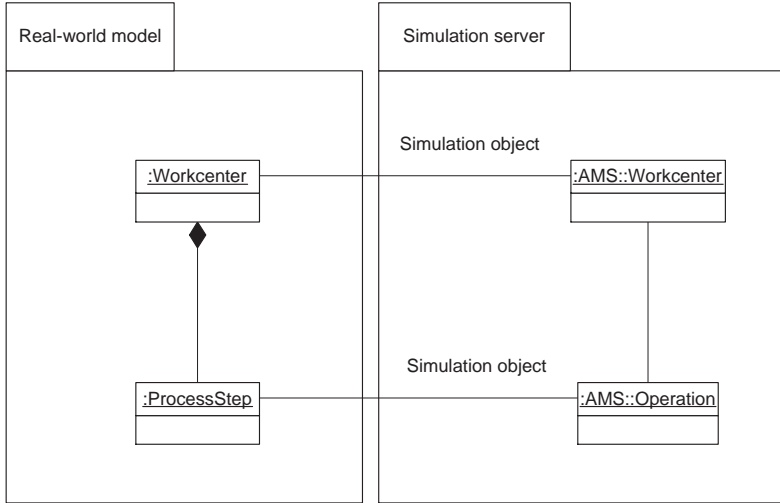


Figure 7.11.: Mapping real-world objects to simulation server objects

model should not be confused with auxiliary simulation objects solely needed because of the underlying simulation engine. All additional objects needed to model reality correctly should therefore be hidden from the end-user.

From the object-oriented point of view a special type of work center is just another class that inherits most of the behavior from normal work centers and adds some functionality needed during the creation of the simulation model. A sequence work center is still a work center that has a name, a description, values specifying the number of tools, etc. The process steps carried out on this kind of work center require a mean process time and also the time of the longest step in the sequence. The way how these additional parameters are handled is described in sections 7.2.2 and 4.3.2. The following sections show how these parameters are used in sub-models created for special work center types.

Creating Sub-Models

Normally work centers and process steps create a single work center and process step object in the simulation model, respectively. An example of this

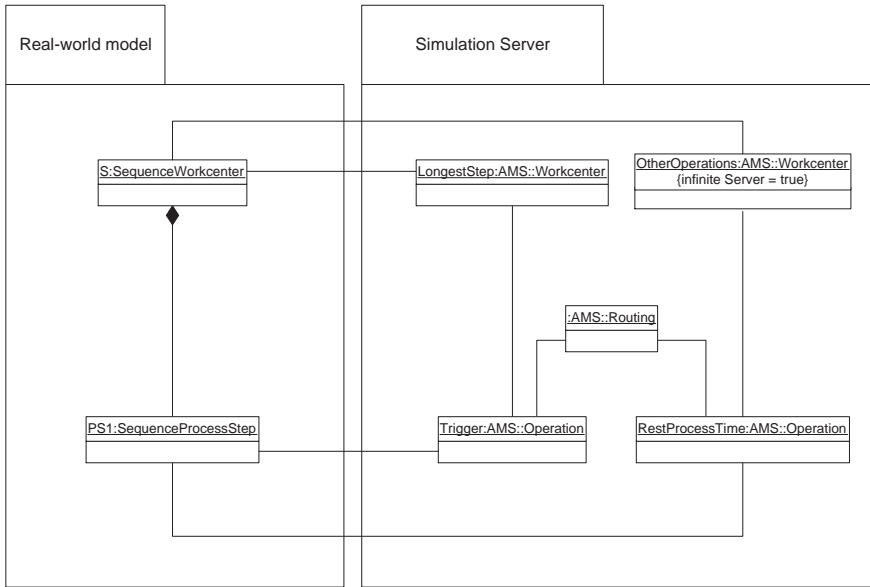


Figure 7.12.: Mapping of a sequence work center.

situation can be found in figure 7.11 which shows an UML object diagram. On the left side a package contains objects from the real-world model residing on the model generator. A **Workcenter** and a **ProcessStep** object each have zero or one associated simulation objects shown on the right side in a package called **Simulation Server**.

To model special work centers, process steps in the simulation model are replaced by a network of **Operation** objects. These additional **Operation** objects can be placed on additional **Workcenter** objects created by the real-world work centers. Thus, work center and process step objects need to work together when creating sub-networks. From the routing point of view it is not of interest if a process step is modeled by a single simulation object or a network of simulation objects. A process step is treated like a black box and always modeled like it was a network of simulation objects. A routing connects the predecessor to the sink and the successor to the source of networks created by the predecessor and successor process steps, respectively.

The class **ProcessStep** needs the following two methods in order to enable the routing to connect sub-networks:

- **getAMS_InputOperation**. This method returns the source operation of the sub-network. This is the input of the black box and is used as the successor of a routing object.
- **getAMS_OutputOperation**. This method returns the sink of the sub-network being used as the predecessor of a routing.

As these methods only return a single simulation object it is obvious that a sub-network can only have one source and one sink operation. This is not a limitation as process steps which use a sub-network can create arbitrary graphs of operations and routings on the simulation server. When a process step is modeled by a single operation the sub-network is trivial: The graph contains only one operation, the source and sink operation; the two methods, **getAMS_InputOperation()** and **getAMS_OutputOperation** return the same value.

Figure 7.12 shows how a simple sub-network made of two process steps is constructed for an object of class **SequenceWorkcenter** which inherits from **Workcenter**. In contrast to the previous figure this example shows two work centers and two operation objects on the simulation server which are assigned to the objects of the real-world model. The sequence work center creates one work center for the trigger operation and another work center for the remaining lead time. The object of class **SequenceProcessStep** which inherits from **ProcessStep** creates two operations and places them on the work centers created by the **SequenceWorkcenter**. These two classes work together: A **SequenceProcessStep** object needs a **SequenceWorkcenter** object which creates the simulation work centers needed.

By using the example of the sequence work center, the general schema of the creation of sub-networks will now be explained: A routing object contains a reference to process step **PS1**, either as the predecessor or the successor. The routing calls the method **createAMS_Objects()** of the process step **PS1** which in turn first calls the same method of the work center that it is part of. The method **createAMS_Objects()** of the work center is overwritten and creates all work centers which are needed in the sub-network, in this case **LongestStep** and **OtherOperations**. Note that the second work center is modeled as an infinite server. This is done to prevent a queue in front of

this work center, as it is just needed to add the remaining process time to the process flow.

After the work centers have been created the process step creates the sub-network of operations on the simulation server. It then assigns the operations to the work center which just have been created. In order to complete the network some routing objects are created and connected to the operations on the simulation server. In the example the sequence process step only needs to create one routing object from `Trigger` to `RestProcessTime` using routing probability 1.0.

The routing which started this process then calls the `createAMS_Objects()` method of the other assigned process step and then creates its own routing object on the simulation server. To do this it needs the operations to be connected. Depending on which side the process step `PS1` is connected one of the following two cases will occur:

- If `PS1` is the predecessor of the routing, the method `getAMS_OutputOperation()` is called which returns the sink operation of the sub-network. This would be the operation `RestProcessTime`.
- If `PS1` is the successor, `getAMS_InputOperation` is called and returns the source of the sub-network. In the example the first operation `Trigger` would be returned.

Besides modeling sequence operations the model generator supports the creation of sub-networks for inspections, work centers performing operations which need an extra chill time, and photo cluster work centers with an arbitrary number of apply and develop stations. The sub-networks used to model these types of machines are discussed in the following sections.

7.2.8. Examples

In the following the sub-models of special work center types that are common in semiconductor manufacturing are presented. Most of these work center types follow abstract patterns that can also be found in other industries, as well.

Modeling a Work Center with an Additional Process Step

In some work centers like ovens or sputter machines, for example, parts need to wait some additional time after the actual operation due to technological

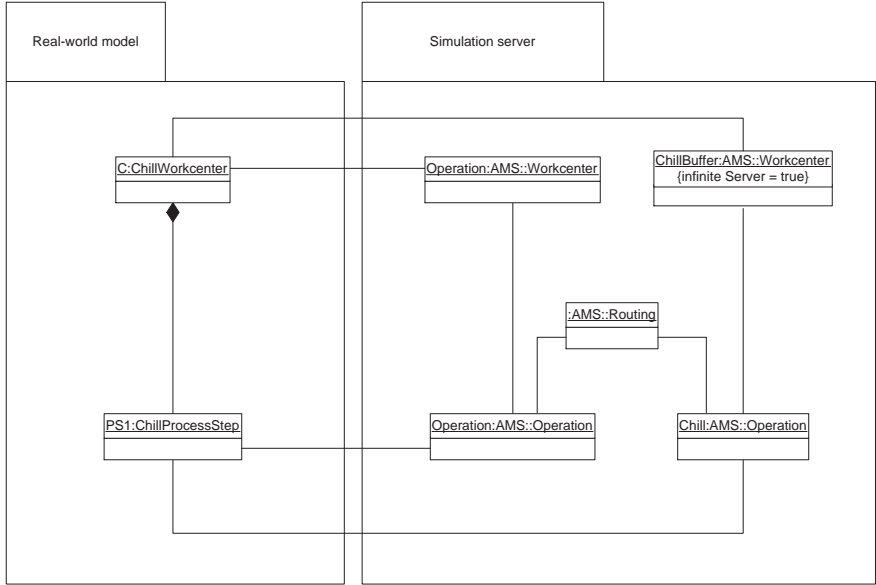


Figure 7.13.: Work center requiring additional chill time after operations.

constraints, e. g. parts need to chill after they have been heated in an oven, parts need to dry after they have been washed, a chemical that was applied needs to rest some time before the next operation, etc. If the additional time — in the following called chill time — can be performed while the work center can already work on new parts, an additional work center in the simulation model is needed. The chill time cannot be modeled as part of the process time of the process step as this would erroneously decrease the capacity of the work center. It cannot be skipped either as this would shorten the raw process cycle time and therefore also the lead time of the whole process. If the capacity of the place where the parts have to wait is reasonably large and can be neglected a special work center type — the chill work center — can be used. Otherwise new operation, work center and process step objects have to be introduced in order to model this situation correctly.

The example in appendix A contains such a work center, the sputter:

Parts which have been processed in this work center have to chill for a least 30 minutes before they can be processed at the next work center. These parts are placed on a table next to the machine; after 30 minutes the parts are transported to the next operation. Assuming that the table is big enough for all parts which are coming out of the sputter in a certain amount of time, the table, or chill work center, can be modeled as an infinite server. This also makes sense for another reason: No parts will ever have to wait at this work center in order to chill.

Figure 7.13 shows the mapping of a chill work center to the simulation objects. This is quite similar to figure 7.12 which shows the mapping of a sequence work center. In both cases the process step is modeled by two operations on the simulation server. The second operation is performed on an infinite server. The difference is the type of work center in the real world where the additional times are specified and denoted. In contrast to the sequence work center process steps for the chill work center need the chill time which is set as the process time of process step `Chill`.

Modeling Inspections

Inspections or so-called *process inspections* are carried out in order to test how far a process fulfills the required quality characteristics [GfQ79]. This can be done by using different methods of statistical quality control [GfQ80, GfQ81]. In the context of automatic model generation the operations used to perform process inspections and the way these operations are modeled are of interest. Tests for quality characteristics do not need to be 100 % inspections. In many cases not all parts are inspected due to various reasons [DS98b]. The sampling fraction is the relation between the size of the sample and the population, e.g. the inspection lot size (definition 1.4.4.3 in [GfQ79]). Depending on the way how batches are handled there are three possible methods:

1. The inspection lot size (the population) is equal to the transportation batch size, i.e. a transportation batch arrives at an inspection work center, a sample of the specified sample fraction is drawn from the batch and inspected.
2. The population is independent of the transportation batch size, i.e. the sample is not drawn from a transportation batch.

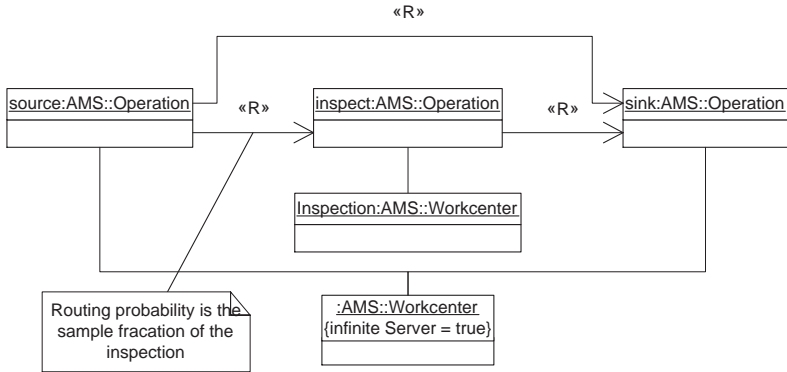


Figure 7.14.: Modeling inspections by routing batches around the inspection operation

- Multiple sampling inspections, sequential sampling inspections, or continuous sampling inspections allow the more elaborate combination of the first two principles, e.g. a first sample is taken from the transportation batches, then another sample is taken from this inspection lot.

When generating a simulation model the mean process times of inspections have to be specified. It is obvious that this time depends on the sampling fraction. In the first case the mean process time can be set as $sample \cdot mpt$ using

<i>sample</i>	the sampling fraction
<i>mpt</i>	the mean process time.

This is exactly what is done in the current version of the model generator. It also reflects the way inspections are performed in the production lines simulated. If the inspection lot is defined like in case 2 and 3 other ways to model inspections are needed.

The second case can be simulated by creating a sub-model like it is shown in figure 7.14¹⁶. The sub-network consists of two operations placed

¹⁶For an explanation of the stereotype «R» refer to section 7.2.8.

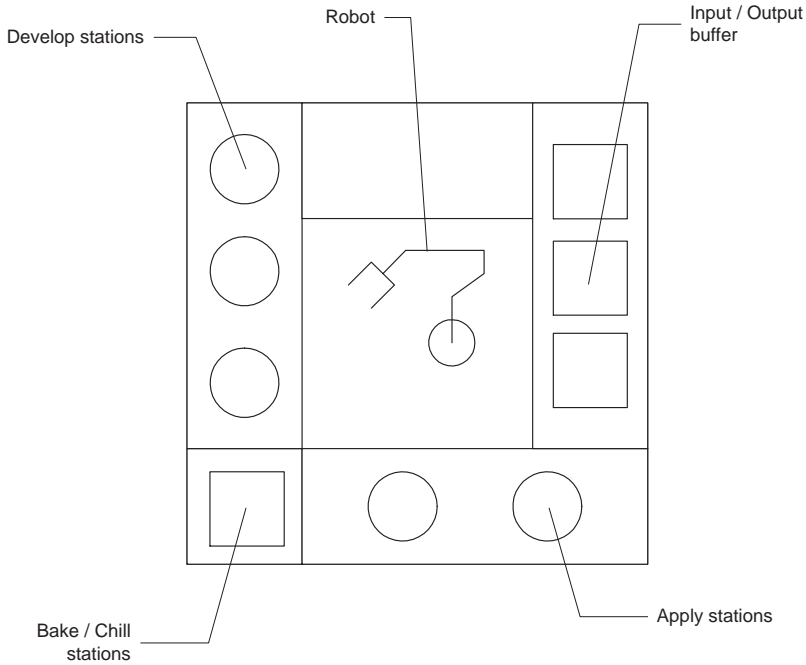


Figure 7.15.: Schematic view of a photo cluster

on an infinite server. These operations serving as source and sink of the network, respectively, have to be included to insert the sub-network into the main network. It is then possible to move batches which do not need to be inspected around the inspection operation which is accomplished by the routing from operation **source** to **sink**.

If the third case is to be modeled in a simulation model for tactical scenarios, first a statistical analysis of the sample plan has to be made. The results then have to be used in a model that combines changing the process time (first case) and routing batches around the inspection (second case).

Modeling a Cluster Machine

Photo clusters often used in the photo cell of wafer productions are complex machines that can perform both apply and develop operations (see Appendix F). Figure 7.15 showing a schematic view of a photo cluster is used to explain the functionality of this machine. In the middle of the machine a robot arm moves wafers from one sub-operation to the next. The input/output buffer (also called nest) is located on the right side. It can hold several batches of normally up to 12 wafers placed in cassettes. The nests are loaded by operators who also supply the information needed, i.e. the batch and wafer numbers and the operation to be performed. When a nest is loaded the robot picks up a single wafer and places it either on an apply or develop station. This depends upon the operation that is to be performed which in turn depends on the state of the wafer in the process flow. Apply and develop stations are actually quite similar: Each station consists of a round plate on which the wafer is placed by the robot. The plate is turned while an arm applies a liquid chemical on the surface of the wafer¹⁷. Depending on the operation the following is performed:

- *Apply operation.* The chemical used is a liquid photo resist. After it has been applied the wafer is picked up by the robot and placed in a bake station where it is heated to bake the resist. After the bake station the wafer is placed into another station to chill. Some operations require two layers of photo resist, then the apply, bake, and chill operations have to be carried out once again.
- *Develop operation.* In the develop station a chemical is applied to develop the photo resist which has previously been applied and exposed. Then another cleaning liquid is applied and thrown off the surface by turning the table very fast. No bake and chill operation is needed in case of develop operations.

¹⁷This is quite similar to a turn table.

Comment: *The maximum throughput of a photo cluster can only be achieved with the right loading strategy. If, for example, an operator fills all nests with batches for apply operations, no develop operations can be started, though the cluster could perform these operations, because the develop stations are idle.*

A wafer needs to run through the photo process of applying, exposing and developing the photo resist many times to construct the different layers needed. Different layers require different types of photo resist and develop chemicals. As the apply and develop stations can only utilize a fixed amount of different chemicals, not all apply and develop stations can carry out each apply or develop operation, respectively. In the example in appendix A the operation *0200 apply layer 1* can only be carried out on the first apply station, whereas *0250 develop layer2* is only supported by the second develop station. A detailed analysis of the photo clusters shows that the internal bottlenecks are either the apply or the develop stations. There are always enough wafer nests, bake and chill stations available, the only time a wafer waits to be processed is before an apply or develop operation. The information that is needed to correctly model the photo cluster must consequently include the time that is needed for each apply or develop operation, the time needed for bake and chill in case of apply operations, and the station on which the operation can be performed.

To explain how the model generator creates simulation objects from the information in the real-world model the example in figure 7.16 is used. It shows an object diagram of the mapping of two process steps placed on a photo cluster work center and their corresponding simulation objects. The objects **PhotoCluster1**, **apply**, and **develop** are parts of the real-world model and are shown outside of the package **Simulation Server** which contains the simulation objects.

To demonstrate different types of operations a single apply and develop operation has been chosen in this example¹⁸. Both process steps do not have any limitation concerning the apply and develop stations, i.e. the apply operation can be carried out on either apply station 1 or apply station 2. **PhotoCluster1**, an object of class **PhotoClusterWorkcenter**, contains the operations **apply** and **develop** which are both instances of class **PhotoClusterProcessStep**. Hence, once again, two classes inherited from **Workcenter** and **ProcessStep**, respectively, have to work together to create the mapping. During the model generation **PhotoCluster1** creates the following simula-

¹⁸Normally a photo cluster would perform one apply and one develop operation for each layer and each product. Assuming a normal load of 15 layers and 10 products 300 different operations are carried out on a single photo cluster. The model generator would then create 2406 simulation objects (6 work centers for the photo cluster and 4 operations and 4 routings per process step)

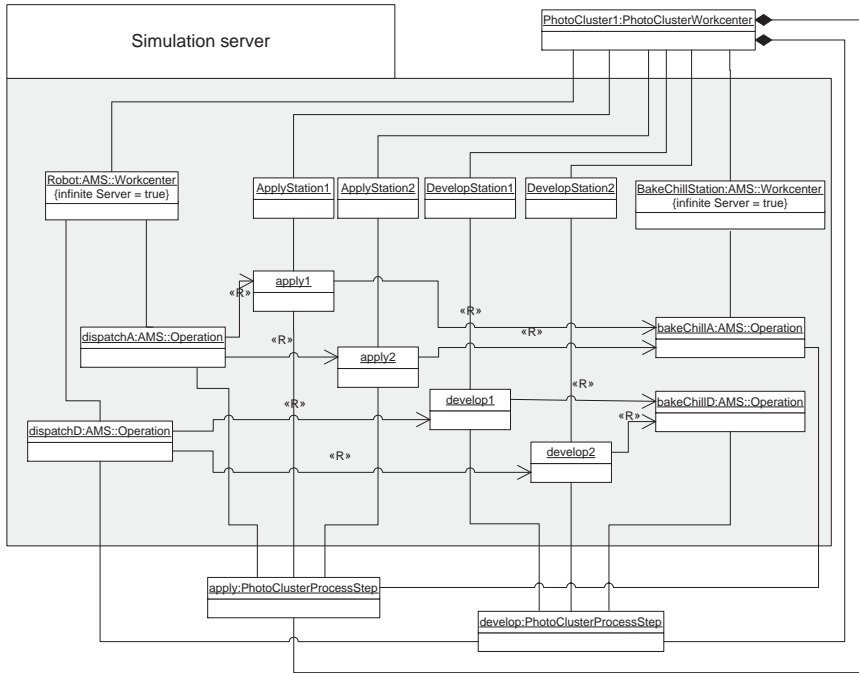


Figure 7.16.: Object diagram of mapping between real-world and simulation objects for a photo cluster work center

tion objects¹⁹:

- **Robot.** This work center is used as the dispatcher to the following work centers and servers as the source of the sub-network which is needed in order to connect it to the outside.
- **ApplyStation1, ApplyStation2.** There is one work center for each of the apply stations in the cluster tool. It is not sufficient to create

¹⁹The objects **ApplyStation1**, **ApplyStation2**, **DevelopStation1**, and **DevelopStation2** are all instances of class **AMS:Workcenter**. For the sake of simplicity this is not explicitly stated in figure 7.16.

just one apply work center and edit its number of tools, because each process step can be limited to a different set of apply stations. This information is specified in the tool-parameter-sheets.

Different types of photo clusters can have different numbers of apply and develop stations. The model generator determines the number of apply stations to be created in the simulation model from the number of parameters of the associated work center type (see section 4.3.2).

- **DevelopStation1, DevelopStation2.** Just like the apply stations one work center is created for each develop station of the cluster tool.
- **BakeChillStation.** This work center is used for the time needed to bake and chill after apply operations. Figure 7.16 shows that the develop operations are connected to this work center, as well, though no bake and chill operation is required. This is done to save another work center which would have been needed as there has to be exactly one sink operation in the sub-network. Develop operations thus use the **BakeChillStation** with a minimum mean process time as the sink of the sub-network. The work center is modeled as an infinite server, because there are always enough bake or chill stations available.

The process steps **apply** and **develop** shown on the bottom of figure 7.16 create numerous operations and routings on the simulation server. The process step **apply** first creates the operation **dispatchA** on the **Robot** work center. From this dispatcher parts are routed to the following operations **apply1** and **apply2**. This is denoted by the stereotyped association between the operations. An association with stereotype «R» is a short form for a **Routing** object between the associated classes or objects. The navigation of «R» associations specifies the direction of the flow of material, i.e. the arrow points to the successor of the routing. In case of process step **apply** the routing leads from the source of the sub-network, **dispatchA** to **apply1** and **apply2** and from there to the sink of the sub-network, **BakeChillA**.

The operations and routings for the process step **develop** are created similarly by the model generator. The only difference is the use of operations which are placed on develop stations (**DevelopStation1** and **DevelopStation2**).

7.2.9. Different Planning Scenarios

Depending on the planning horizon different models need to be created for simulation studies. This is accomplished by the automatic model generation which simplifies the maintenance of models and broadens the area in which they can be used.

One difference between simulation models becomes apparent when modeling work centers and the process steps being carried out on them. For short term planning stricter constraints apply to mapping products and operations to work centers than in long term planning.

The EPOS model generator can group *operational* work centers by collecting all process steps of these work centers and placing them on a new, single *tactical* work center. The information which kind of model is to be created is stated in the simulation request (see table 7.1). Depending on the return value of the method `createTacticalModel()` a normal (operational) or tactical model using work center grouping is created.

Grouping of work centers can be integrated seamlessly into the model generation by using a simple trick: When associating a process step to a work center in the simulation model the method `getPlanningWorkcenter(boolean tactical)` of a process step is used. Setting the parameter `tactical` to false lets the method return the work center that the process step is associated to in the real-world model. Specifying true skips this work center and returns the tactical work center. If a work center is not grouped into a work center group, the work center itself is returned. That way all process steps located on operational work centers will be placed on the tactical work center, if one exists. This even works transitively: A tactical work center can also be placed in a work center group.

Parameters of the tactical work centers can be set in two ways: Firstly, a parameter which is defined for the tactical work center is directly used in the simulation model. Secondly, if a parameter of the tactical work center is omitted, it is calculated as the average of the other work centers in the group. If no entry for the number of tools is supplied, the sum of all associated operational work centers is calculated for each specific week.

7.2.10. Multi-Process Production Lines

Some production lines carry out processes where parts are split or joined. An assembly line is an example of a production where different parts are

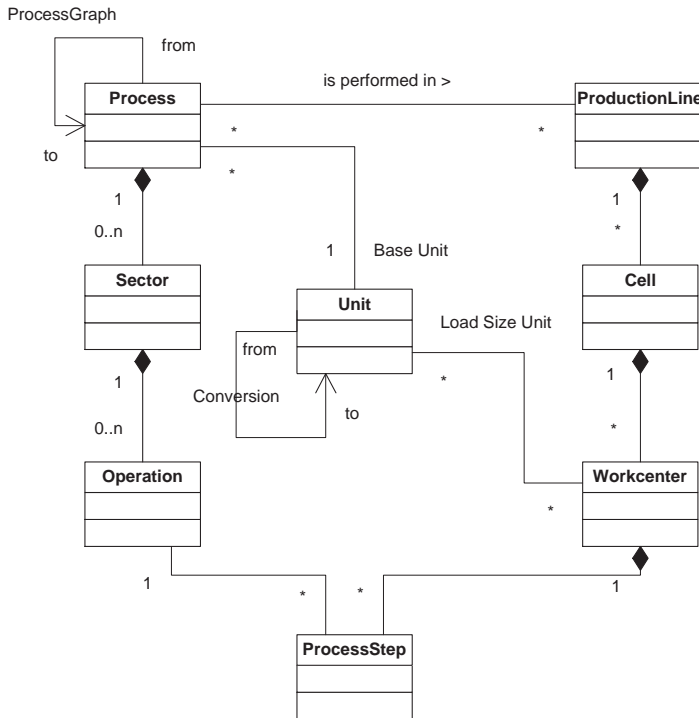


Figure 7.17.: Relationship between processes and production lines and the associated units

joined together in an assembly process. In chemical productions different parts are often joined together. There are also processes where parts are split into other parts, an examples is an oil refinery where the splitting of crude oil produces gasoline, light or heavy fuel oil, bitumen, and gas [Sch93]. In the production of read/write-heads split processes occur in the so-called slider lines where wafers are cut into smaller units like rows or sliders (see section F.3).

Figure 7.17 shows the relationship between processes and production lines in the EPOS model (see also section 4.3.1). Each process has exactly one

associated base unit. All process steps which are performed in this process²⁰ are specified in the base unit of this process. The work center on which a process step is carried out also has an associated unit, the load size unit. This unit is used in the parameter collection to specify the load size of the work center. If the load size unit of the work center and the base unit of the process differ, the batch size of the work center is converted to the base unit of the process by using the conversion factor²¹. The possibility of specifying a different unit at the work center has been introduced for ease of use during the parameter collection. Different machines use different types of loading mechanisms. The engineer who has to enter the batch size should be able to do this using the unit that he is accustomed to.

The diagram also shows that it is possible to have two or more processes with different base units in the same production line which is used to model splitting or assembly productions. If a production process splits parts, a process for each unit that parts appear in has to be set up. Slider production lines in which wafers are cut into quads and rows are an example: The first process uses the unit *wafer* as its base unit. Then another *quad* process and at last a process using the unit *row* is specified to be carried out in the production line. The order in which these processes are to be performed needs to be specified in the table **ProcessGraph**.

During the model generation processes having different base units have to be integrated into one simulation model. The simulation server does not have any means to specify parts in different units, though. Nevertheless, the traditional means to model product structures — the bill-of-materials which can be described by a Gozinto graph (see section 4.3.3) — is available in the simulation server and can thus be utilized to model different base units. But why is the bill-of-materials of the EPOS model not used directly? The answer is that the possibility of generating simulation models with different base units greatly simplifies the maintenance of the model. To explain this the difference between the BOM and the units has to be made clear (see table 7.4): The bill-of-materials is a relation between different products, whereas units are used to describe relations within the same product. This definition directly applies to the situation in slider lines. A *Taurus* product that is started into the process, for example, leaves the production line as the same product, it just has changed the level of aggregation (from one wafer to

²⁰Note the links from class **ProcessStep** to **Operation**, **Cell** and then **Process**.

²¹The model generator calculates the correct conversion factor, if the conversion table specifies an arbitrary route from the source unit to the destination unit.

Area	BOM	Unit
Definition	Relation between different products	Relation between units of the same product
Granularity	Defined for every product	Defined for a product size
Maintenance	For each new product one product for every level of aggregation has to be created. Then the factors between the different levels have to be specified in the BOM.	Conversion factors are defined for product sizes. If a new product uses an already defined product size no specification is needed.
Usage	Assembly processes, processes that split products into different products	Processes that just change the level of aggregation, not the product

Table 7.4.: Differences between BOM and units

a few hundred rows). During the model generation a new product for each base unit in the production line has to be created and the Gozinto graph on these products has to be generated on the simulation server. Without the possibility of using different units in a production line this tedious work would have to be done manually.

To create the Gozinto graph first products for all combinations of base units and product groups to be simulated have to be created. This is automatically done by the way product groups create their corresponding simulation objects: The method is called from the process step which passes the unit of the associated process to the product group which maintains a table of all simulation objects indexed by the unit (see section 7.2.4). Production lines with just one unit are therefore just special types of multi-process production lines. After all process steps (and by this all products) have been created the **Need** objects are generated on the simulation server. To initialize a **Need** object the conversion value has to be calculated. This can be done by using the knowledge about the order in which the processes are carried out. The simulation model then consists of different routing components which are separated by the units of the processes. The secondary demand for the first processes is calculated from the primary demand of the later units and the information stored in the **Need** objects.

7.3. Automatic Simulation

After the simulation model is created the actual simulation can be started on the server. Whereas it is always possible to create simulation models and exchange these via files of some kind of format, the control over the simulation run needs a tight integration of the model generator and the simulation server. The EPOS simulation server offers such integration via its CORBA interface. This permits to start a simulation run and to read the results from the server.

Phases 1–3 introduced in section 7.2.1 on page 171 have been covered in the previous sections. This section deals with phases 4–7, i.e. the simulation of the model and the preparations which are needed before the actual simulation run can start. Then the collection and aggregation of the simulation results is presented. The section closes with the description of the calculation of line profiles.

7.3.1. Simulating the Model

The basis for a complete simulation study is the volume plan specified in the simulation request. It specifies the planning horizon and the demand for each week of the plan. The simulation server calculates performance values for the stationary states. Time dependent phases or parameters cannot (yet) be simulated on basis of the methods used (see sections 3 and 6.1). Changing demand and parameters can be simulated as different scenarios, though. It is assumed that the simulated model reaches a stationary state within the time frame of the scenario.

A complete simulation run consists of many different scenarios, one for each week of the volume plan. Thus, before a single simulation run can be started the time-dependent parameters of the model, i.e.

- the demand specified in the volume plan,
- the number of tools of a work center,
- the routing including the amount of rework and scrap,

have to be adjusted for the specified week. To accomplish this the model generator first resets all product groups by setting the demand to zero. Then the week's demand is read from the volume plan and set in the simulation model. To set the correct number of tools first all work centers are scanned

for changes which are updated in the simulation model. The number of tools affect the local capacity of work centers (see section 7.2.5). That means that all routing objects pointing to these work centers have to be adjusted, as well. At last changes in the routing, scrap, and rework have to be updated.

After all time-dependent parameters have been changed in the simulation model, finally the simulation can be started by calling the `calculate()` method of the model object on the simulation server [Kle00a]. The simulation server first checks the consistency of the generated model²². Then the object model is transformed into a number of matrices which are used to calculate the needed performance values (see section 6.1).

7.3.2. Saving the Results

When the calculation of performance values is finished the results of the simulation have to be read from the simulation server. The place to store the performance values is the central database:

- A relational database is well suited for the management of large amount of structured data.
- If simulation results are stored in the same database with the structure and the parameters of the model, results and parameters can easily be combined by the use of SQL queries.
- All clients which can use an ODBC or JDBC driver can utilize the simulation results. This makes it possible to employ elaborate systems for reporting of simulation results (see chapter 8).

The model generator therefore loops through the simulation objects, reads all performance values and inserts them into the central database. The advantage of this procedure is the possibility of drilling down into the data as results are calculated with fine granularity. While the simulation normally runs off-line the results of the last simulation runs are always available on-line. The disadvantage — storing all performance values of all simulation objects generates a mass of data in the central database — does not create a problem. Today's databases can easily handle the amount of data generated. Moreover, simulation results are marked with an expiration date after which

²²This step could actually be skipped as the process of model generation ensures the creation of consistent models.

they are automatically deleted by the model generator, i. e. there are only a few sets of all results stored at the same time.

As one would expect, the structure of the simulation results is similar to the structure of the model itself. Figure 7.18 shows an UML class diagram of the tables used. The center is class **Simulation** which has already been presented in section 7.1.3 (see figure 7.5). A **Simulation** object represents a simulation run and consequently all results need to be associated to such an object. The abstract class **SimulationResult** is associated to the classes **Year** and **Week** expressing the fact that each simulation result belongs to a scenario for a specific week and year. On the bottom of the diagram the **Main** package is shown containing classes from the EPOS core model, namely **ProductionLine**, **Workcenter**, **ProductGroup**, **Unit**, and **ProcessStep**. Each class of the simulation results is associated to at least one of these core classes. The class **WCResult** is associated with the class **Workcenter**, because each **WCResult** object (tuple) represents the result for one work center in the model. The class **PGroupResult** depends on the classes **ProductGroup** and **Unit** as one simulation object for each combination of product group and unit is created. This differentiation is needed to enable the simulation of multi-process production lines (see section 7.2.10). One class is even associated with three core classes: **WcPGroupResult** is used to store the combined work center and product group results. Once again, the association to the class **Unit** is needed for multi-process production lines. The other classes, **ProdLineResult** and **ProcessStepResult**, use the same schema presented.

To store the results of special analyses like line profiles (see section 7.3.3) the classes **Analysis** and **AnalysisValue** have been introduced. Each simulation run can have an arbitrary number of additional analyses. The name of the analysis and a comment describing it are stored in an **Analysis** object. Mathematically an analysis is a relation $A \subseteq \mathbb{R} \times \mathbb{R}, |A| < \infty$, and thus can be used to store the data for a two dimensional chart. The member **series** is a string which can be used to separate an analysis into different relations. Using a reporting system the string **series** can be used to draw a different line of set or bars for each sub-relation (see figure 8.11).

Aggregating Sub-Models

While generating a simulation model some objects in the real-world model are modeled by subsets of additional simulation objects. These simula-

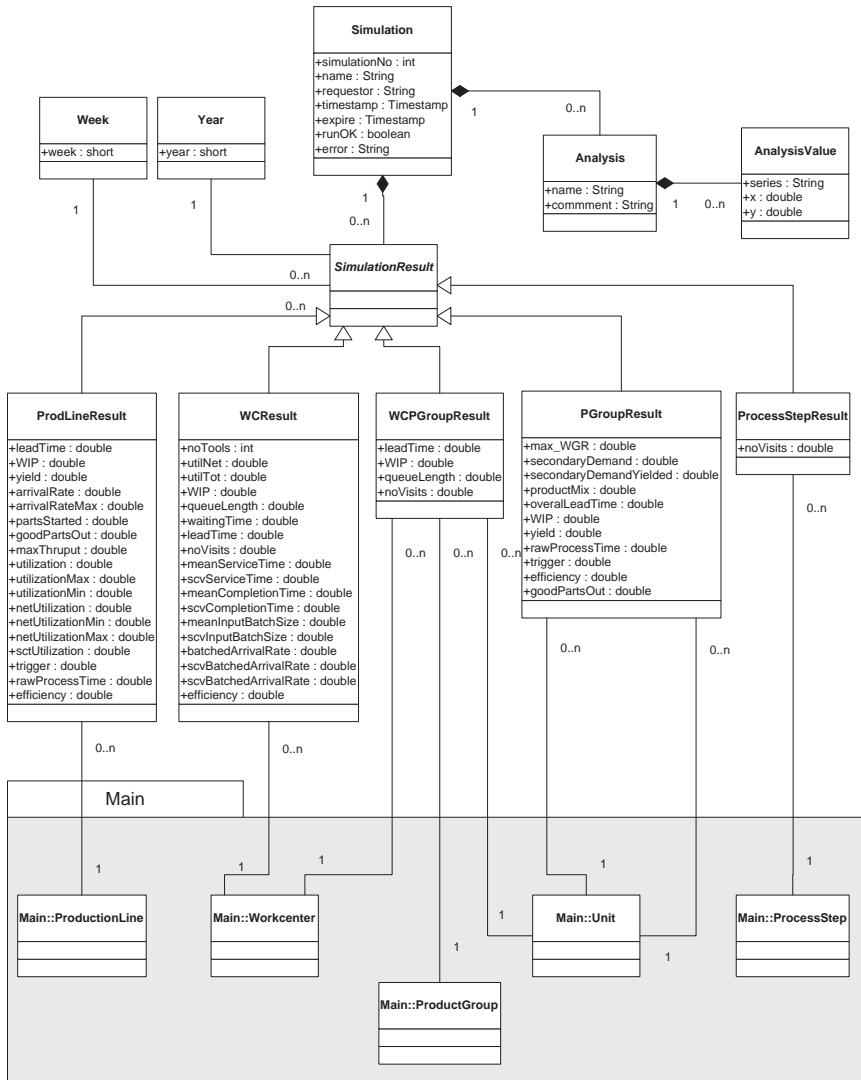


Figure 7.18.: Results of simulation run

tion objects do not have any corresponding objects in the real-world model. Moreover, results of these auxiliary objects should not be stored as simulation results, but should be aggregated to results of the real-world objects for which the sub-network was created. This is done in order to hide complexity from the user of the system.

To incorporate this feature the results are not read directly from the simulation objects. Instead, a method of the real-world object is called which reads the performance values of the simulation objects and normally just returns this number. In the case of special work center types this method is overridden in order to aggregate the results of the auxiliary objects. The following list shows some of the aggregation functions used:

- *Utilization.* For sequence work centers and work centers requiring a chill time only the first work center is considered. In both cases the second work center, for the remaining lead time or chill time, respectively, is modeled as an infinite server which by definition has a utilization equal to zero. Photo cluster work centers return the maximum utilization of any apply or develop station which is the bottleneck of the machine.
- *Work-in-process.* The work-in-process is calculated for all special work center types in the same manner: The work-in-process results of each auxiliary simulation object is summed up to obtain the aggregated performance value for the sub-network.
- *Lead time.* The lead time of the sequence work center is the sum of the first trigger work center and the second remaining lead time work center. The same is performed for work centers requiring an additional chill time. This is done as in both cases parts have to pass through both sub-operations. The lead time of the photo cluster has to be calculated differently as some parts are not processed on each work center of the sub-network for each operation. This is only true for the dispatcher and bake/chill work center at the source and the sink of the sub-network, respectively. Thus, firstly the sum of these two work centers' lead times is calculated. Then the lead time of the remaining apply and develop stations is added. This time is calculated using the weighted sum of the station's lead time.

Converting the Work-in-Process for Multi-Process Production Lines

Some production lines perform processes that use different base units (see section 7.2.10), i.e. parts which are produced in such a production line can be in different states or are specified in different units. The simulation server has no means to describe parts in different units, though. This means that the work-in-process of the production line which is calculated in the simulation server sums up parts in different units.

This makes sense in the following scenario: A robot sprays paint on doors or the whole body work of a car. The product structure of the simulation model would specify that two, four, or five doors would be needed for one body work. The outcome of the simulation would be a work-in-process of x parts. These x parts would either be doors or body works. An analysis of the products being sprayed at the work center could show how many doors and how many body works would make up the work-in-process. The total work-in-process of the production line would consist of different parts regardless whether they might be doors, body works, or maybe tires.

In the case that a production line produces processes with different base units this information can be used to present the work-in-process in one unit. The difference is that units are used to distinguish different forms within the same product whereas the example above specified two different products, doors, and body works. If a multi-process production line is to be simulated different units of the same product have to be modeled as different products in the simulation model (see section 7.2.10). To calculate the total, aggregated work-in-process at the work center or in the whole production line the functionality of the simulation server cannot be used. Instead, the model generator reads the results separated by product groups and units and converts all values into one unit before aggregating the performance values. The unit chosen is the base unit of the last (sink) process which is performed in the production line.

7.3.3. Calculating Line Profiles

A line profile is a chart that shows typical performance values like work-in-process, lead time, or efficiency over the possible range of utilization of a production line (see figure 8.11 in section 8.5.1 for an example of a lead time profile). To calculate such a profile a complete simulation run has to be carried out for each point of the profile. Using the queueing network

analysis a fast calculation of line profiles becomes possible, whereas discrete event simulation requires many runs and a statistical analysis for each point.

After the simulation for all periods of the volume plan is finished the calculation of the line profile is started, if it is requested in the simulation request. The simulation request also specifies how many runs should be started and what the interval of utilization should be. One of the following two ways to calculate the profile must be chosen for the request:

1. *AMS line profile.* This demands that the calculation of the whole line profile is to be left to the simulation server which offers this functionality. This is the faster method as the simulation server can save overhead, because it does not need to recalculate every piece of information. The calculation is started once and the result of the whole line profile is returned to the model generator.
2. *EPOS line profile.* This method leaves the control of the profile calculation at the model generator. It prevents the re-use of already calculated performance values and is therefore not as fast as the first method. The method is needed, though, if a profile for multi-process production line is to be calculated, because the simulation server has no means to differentiate various units in the model (see section 7.2.10).

As the AMS line profile is computed in the simulation server [Kle00a] only the second method is described here. To simplify the calculation the product mix of the whole volume plan is used. Thus, first the average demand per week is calculated and a simulation run is started on basis of that demand. This yields the utilization of the bottleneck of the production line. Using this value the demand is then scaled linearly from 0 to the bottleneck utilization and a simulation run is started for n equidistant points. After a single run has completed the performance values of the production line, namely work-in-process, lead time, and efficiency are read from the simulation server and stored in the database table `AnalysisValue`.

7.4. Processing Inconsistent and Incomplete Data

In section 2.7.3 the need to process inconsistent and incomplete data is discussed. The following section deals with mechanisms used to process data that is either inconsistent, missing, or incorrect. First some principles

of the automatic model generation are discussed, then the different types of faulty information and ways to cope with it are described in detail.

7.4.1. Principles

When a simulation model is to be created in a fault tolerant way different techniques to surpass the arising problems can be used. The EPOS model generator uses straight forward, but effective methods. For missing parameters default values are substituted. If data is inconsistent various assumptions are made in order to solve contradictions. The principles behind the correction mechanisms are quite simple:

1. A simulation model should be created in any way.
2. The influence of any *additional* activity needed to generate a computable model should be kept at a minimum.
3. Every *additional* activity must be documented.

While the first principle expresses the need to create a simulation model at any cost the other two focus on the way a simulation model is created.

The second principle implicitly states that on the basis of incomplete and inconsistent data some *additional* activities are needed in order to create a model that can be simulated. These activities are to be performed in such a way that faulty or missing data influence the outcome of the simulation to a minimum extend. If this were the case, information about the direction and the amount of influence is needed. But exactly this information is missing or faulty. This has consequences on the way how default values are chosen. If the process time of an operation is missing and must be specified, for example, it should be set to the lowest possible value.

The third principle demands that every additional activity performed in order to *repair* faulty data must be documented. This is quite important as the outcome of any simulation can only be interpreted in conjunction with the assumptions that the model is based on. Section 7.4.4 describes the event logging in detail.

7.4.2. Incomplete Data

Incomplete data means that part of the information needed to create the simulation model is missing. In this case the missing parameters can be

initialized by reasonable default values. The following list shows where incomplete data can occur and what type of default value is used.

- *Work center.* The available number of tools can be missing. During the generation the number is set to 1 which enables a simulation. If operations on this work center can also be carried out on other work centers, all parts are routed to these work centers. Otherwise the simulation will be performed assuming that one tool exists. In that case a severe warning is issued.

The reliability and the mean down time are calculated from the following parameters: breaks, pfd, lunch, engineering time, set-up time, monitoring time, preventive maintenance, mean time to repair, mean time between failures. A missing parameter is replaced by zero meaning no time is needed for the particular activity. A missing MTBF parameter can be substituted by setting it to a positive value and MTTR to zero. No time is lost therefore due to machine down times.

Moreover, the coefficient of variation for the down time could be left out. The automatic model generation assumes a deterministic tool by setting this parameter to zero.

- *Process steps.* The required parameters for process steps are load size, mean process time, and its coefficient of variation. Depending on the type of load size to be used in the simulation model the value is taken either from the work center that the process step is placed on or from the process step itself. If the minimum or maximum load size of the work center are missing an event for the work center is created, and a batch size of one is used as default. The same is done for the process step if its actual load size is missing.

A missing mean process time defaults to the smallest possible value; the coefficient of variation is assumed to be zero if missing.

Special work center types like inspections, photo clusters, or sequence work centers need extra parameters which can also be missing. If the sample rate for an inspection is not specified, 100% is used as a default. The longest step of a sequence work center is replaced by the mean process time of the process step. Additional chill time on a work center that requires chilling after the process time is set to the smallest possible value.

- *Product groups.* Only current product groups are created when the simulation model is generated. If a positive demand for products in product groups which are not marked as current is found in the volume plan, the demand cannot be set correctly. A severe warning specifying the demand is issued and the simulation proceeds without setting the demand. If no demand is specified in the volume plan for any product of a product group that is being simulated, it is assumed that the demand for the product group is zero. If the volume plan does not specify any demand for a given period, a warning is generated and the period is skipped.
- *Routing.* The routing in the real-world model has just one parameter, the routing probability. By integrity checking it can be assured that this parameter p satisfies the constraint $0 < p \leq 1$. When setting the routing probability in the simulation model the *local* capacities of the work centers of the successor process steps are taken into consideration, as well (see 7.2.5). The calculation of these capacities requires the work center's number of tools and all of its process steps's process times and load sizes. If any of these required parameters is missing the default values which have been described above will be used. A missing number of tools always results in a work center capacity equal to zero. Thus, the number of tools are assumed to have higher priority than process times or load sizes.

7.4.3. Inconsistent Data

Inconsistent data means information that is not in accordance or that contradicts itself [Cob89]. In most cases information becomes inconsistent by associations between two or more objects. It also occurs, however, that member variables of a single object contradict themselves. Many sources of inconsistency can be eliminated by using different techniques before a simulation model is created. Some of these techniques are already active at the database level:

- *Referential integrity.* Referential integrity can be used to ensure correctness of inter-relational dependencies (see [Vos00b], [Ul188], [HS00], etc.). A process step, for example, can only be created for an existing work center, a routing only from an existing operation to another existing operation.

- *Model design.* A well designed model does not permit any redundancy which could be a cause for inconsistencies.
- *Check constraints.* This database feature can be used to control dependencies within a single tuple in a relation. The effective working hours of a work center are determined by several parameters which are entered by different persons during the data collection. A check constraint could be used to ensure that the time used for engineering, maintenance, and manufacturing activities is less than 24 hours per day.
- *Trigger.* When inserting, deleting, and updating tuples in a relational database triggers are often used to assure integrity. Triggers are used, for example, to control if the unit of a work center is defined in the same company as the work center itself. Neither referential integrity which only ensures that a unit exists, nor check constraints which are limited to tuples themselves can be used perform this check.
- *Queries.* To locate inconsistencies database queries can be utilized. They can either be used directly to find contradicting data or they are implemented in triggers.
- *User interface.* A well designed user interface prevents the user from entering inconsistent data. An example is the use of a combo-boxes from which users can only choose valid parameters.

However, there are dependencies which cannot be controlled during data collection. The main reason for this is the distributed management of data. Certain parts of the real-world model must be managed separately. The next section deals with inconsistencies arising when the routing is joined with the assignment of process steps to machines.

Routing versus Process Steps

The way the routing is stored in the database is different from the way it is used during the simulation. These differences are described in section 4.3.5. The main benefit of this design is the possibility of managing process flows (routing) and machine dedications (process steps) independently. Therefore different persons can work in parallel at the creation of the real-world model. Otherwise the definition of the process flow has to be done after specifying

machine dedications, and deleting a process step would cause the deletion of a routing object²³, which could lead to the destruction of the process flow.

If inconsistencies between the routing and the assigned process steps become apparent during the model creation, the missing process steps are created and assigned to the so-called *dummy work center* (see section 7.2.2). All parameters for these process steps and the dummy work center itself are initialized by default values. The corresponding simulation objects are solely created to enable but not to influence the simulation. Thus, the reliability of the dummy work center is set 1 and it is modeled as an infinite server. Each process step is assigned a minimum positive process time and no variance. During the actual generation of the simulation model no special actions have to be taken. The dummy work center is treated just like a normal work center. Results cannot be written back to the database as the object is transient i.e. it only exists during the simulation. An event stating the number of operations on the dummy work center is created, though.

Solving inconsistencies between routing and process steps relies on the principle that the specification of the routing is more important than the specification of process steps. Inconsistencies resulting from an existing **Routing** and no **ProcessStep** object is therefore interpreted as a correctly specified routing and a missing process step. The reason for the preference of the routing lies in the fact that it can normally be imported from existing data sources like shop-floor-control systems.

Inconsistencies may also have advantages: The differences between the routing and the process steps can be used to control either part of the database. Assuming the routing is correctly specified a report comparing the number of operations which are part of the process flow and the number of specified process steps can be used to control the process of data input (see section 8).

Routing Probabilities

The sum of the routing probabilities of routes leaving an operation must be equal or less to 1, i.e. the difference between 1 and the sum, the scrap ratio, must not be negative. In this case, the model is obviously inconsistent and a severe warning is issued. A report that detects these operations can be used to fix the model.

²³In case the cascaded deletion in the definition of the foreign key is enabled.

Sinks and Sources

It is assumed that the process flow of each product group has exactly one source and one sink operation. These operations are determined during the analysis of the process flow (see section 7.2.5). If a model contains more than one source or sink operation per product group, it is still generated correctly, but a warning specifying the number of sources or sinks, respectively, is generated. The name of each source or sink operation is also specified in a normal event. This information helps to find problems in incorrect process flows.

Work Center Down Time

The down times of a work center are an example of possible inconsistencies within single object. As the times are edited by three different departments (see section 5) it is possible that the sum of all down times is greater than 24 hours per day. This can easily be prevented by using check constraints in the database. An incorrect figure entered by one user could prevent the correct input of another user. As this situation is not desirable no check constraint are used for the work center table. However, a view showing possibly incorrect values is used in the tool-parameter-sheets application. During the model generation a severe warning is issued and the reliability is reset to 1. Thus the model can be simulated yielding the possibility of using results of other work centers.

7.4.4. Event Logging

One of the principles of the automatic model generation states that all changes and activities to enable the simulation of a model have to be documented. This is done by creating events in the **EventHistory** table (see figure 7.5). Each event has an associated priority or level of severity which can be used to sort the events in the order of their possible effect on the outcome of the simulation²⁴. Warnings containing detailed information about any modeling problem and the outcome of the simulation can both effectively be used in the debug and validation process. After successful simulation runs

²⁴While it is not possible to define a strict order of importance of events, some are clearly more important than others. A large number of missing process steps normally affects the simulation more than a missing squared coefficient of variation of the down time of a single work center, for example.

Sev.	Event	Description
1300	Model problem	A severe problem, e. g. a process step performed on a single work center which does not have any tools
1400	Missing operation information	The number of process steps which are placed on the dummy work center
1500	Demand, but no current product	There is a positive demand in the volume plan for a product group which is not simulated.
1700	Sink	A process step does not have any successor and is marked as a sink.
1701	Source	A process which does not have any predecessors is marked as a source in the simulation model.
1750	Routing problem	Routing probabilities adding up to more than 1.0 or more than one sink or source, respectively
2000	Missing number of tools	The number of tools of a work center is not specified.
2010	Missing batch size	The batch size of a process step or work center is not specified.
2015	Batch size adjusted	The batch size had to be adjusted.
2016	Batch size below 1	After scaling a batch size is smaller than 1 and had to be adjusted.
2017	Mean process time adjustment	Due to a batch size below 1 a mean process time had to be adjusted.
2020	Process time missing	The mean process time of a process step is not specified.
2030	Reliability incomplete	Some value needed to calculate the reliability of a work center is not specified.
2040	Mean down time incomplete	Some value needed to calculate the mean down time of a work center is missing.
2100	Work center variance missing	The coefficient of variation of the process time is missing.
2110	Process time variance missing	The coefficient of variation for the process time of a process step is missing.
2130	Down time variance missing	The coefficient of variation for this work center is missing.
3000	General assumption	General assumptions made during the automatic model generation
6000	Scaling	The batch size is scaled to another unit

Table 7.5.: Events generated during model generation.

the results are always presented together with the list of events serving as a complete and condense list of assumptions.

7.5. Summary

The automatic model generation is a very important part of the integrated simulation. EPOS contains a very robust model generator which can create simulation models for a vast area of questions in production management. Methods for the treatment of inconsistent and incomplete data proved to be invaluable in constantly changing environments by facilitating the debugging and validation processes. Using automatic model generation and simulation it is always possible to use the results of a current simulation. It still cannot replace the validation of the simulation model, though. The simulation results must not be used as a basis for control strategies in the operational control of the production without a final check of the production planning department. But it is obvious that the automatic model generation helps to reduce the time that is needed for the validation dramatically, thus the effective use of simulation in the operational planning becomes possible.

Chapter 8

Reporting

EPOS Reporting assures that users can easily access the information they need. Users in this case are engineers who just want to see how their tools rank in the latest simulation results, capacity planners who need surveys on the overall capacity including bottleneck charts for different scenarios, manufacturing managers who want to know whether they will run into problems, EPOS administrators who need to check consistency of the simulation models in the database, etc. Most of these tasks are performed over the company's intranet.

In general, two different approaches are possible, static and dynamic presentation of information. Static reports are generated on a scheduled basis, so that the reports do not necessarily represent the current state of the database. The advantage is that they can be scheduled at night time to balance the load on the database server. Dynamic reporting involves a database query that is performed in the moment the user asks for the information. This requires more efforts to implement and might generate more load on the database in unfavorable circumstances. In order to combine the benefits of both approaches, EPOS provides both kinds of reports depending on the type and use of the report as shown in the following sections. *Integrated simulation* requires the following key features for reporting:

- Static and dynamic reporting
- Pull instead of push techniques

- Integration in IT infrastructure

How this is realized is shown in the following sections. The next section focuses on the technical background of the reporting system followed by sections giving a description of the reports provided by EPOS. As the simulation results for the IBM production lines are confidential either the results of sample models are shown or the part of the charts are removed so that absolute values cannot be seen. However, the structure of the reports becomes obvious in any case.

8.1. Deployment

Figure 8.1 shows the deployment of the EPOS reporting subsystem. The figure clearly shows that the clients access the reports via the intranet with a web browser. Thus the complex architecture providing the web reports is hidden. All data to be reported can be found in two databases, namely the relational database and the Notes Database. The information of both databases can be accessed via a single web site, the EPOS intranet site.

The Notes database contains the tool-parameter-sheets and information on the EPOS project. The domino tasks allows the publication via the intranet (HTTP).

The relational database contains all planning parameters and the simulation results. The information can be accessed via dynamic or static reports.

Static reporting is realized by a reporting engine. In the case of EPOS this is Business Objects (BO)[Bus]. The BO designer allows to define the design of a report. Data is taken via ODBC from the database and the designer can interactively create categorizations, aggregations, sorting, rankings, etc. If the report is finished, it is sent to the BO Broadcast Agent. This is a scheduler that takes the report definitions, queries the database and outputs the current report on a web server. Dynamic categories are automatically realized as linked HTML frames.

Dynamic reporting is realized with the help of the PHP Hypertext Pre-processor [PHP] — a server-side HTML-embedded scripting language — and the Apache Web Server [Apa] on a Linux operating system. A web page with input forms allows the user to select some data, for example a product. The data from the form is sent to the web server where the PHP module takes the parameters, initializes a database query, and generates HTML output that is distributed by the web server.

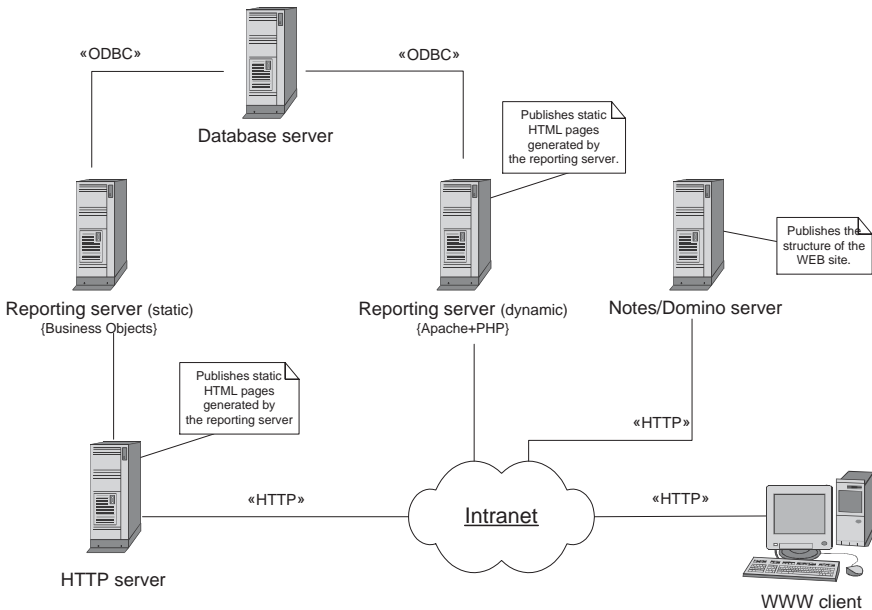


Figure 8.1.: Deployment of the reporting subsystem

PHP is used by embedding the PHP code in HTML files. The embedded code can output text, perform some calculations, or query a database, for example, and is enclosed in special start and end tags. PHP code is executed on the server. The client just receives the results of running the script. Apart from common programming language features like variables, functions, etc. PHP offers special support for database access and other services using protocols such as IMAP, SNMP, NNTP, POP3, HTTP, raw network sockets, etc.

8.2. Types of Reports

Static and dynamic reports each have their own advantages and disadvantages:

- *Static reports.* Today, there exists a huge variety of tools that allow the automatic generation of HTML pages from databases. An approach suitable for advanced users is the structured presentation of all simulation results. This allows the simulation experts to browse through the complete result set and to look for problems, for example.

Another possibility of static web pages is the generation of standard reports distributed by a web server. In this case users get a link directly to their favorite reports so that they can access them with a single mouse click.

In both cases the users have to access the reports themselves instead of getting them sent. This reduces redundancy and thus avoids versioning conflicts that arise when reports are sent to other employees.

The advantages of static reports are that the load on the database can be determined exactly and access to the reports is fast as the charts are simply loaded and no database query is executed.

The disadvantage is that many different views have to be generated and stored on the web server in advance. The charts are not up-to-date if the reports are scheduled only at fixed times.

- *Dynamic reports.* These can provide access to standard reports or the whole set of results, too. But in contrast to static pages, the user is offered more flexible browsing capabilities and might even change the granularity of aggregated values, e.g. mean values for different time intervals or a zoom into a subset of results.

The advantages are that up-to-date data is fetched from the database and only reports needed are generated. The disadvantage is that it takes longer till the user gets the results as data is fetched on the fly. Moreover, each time a report is needed a database query is started. Thus the load on the database server can only be estimated in advance and depends on the number of users.

In the following the EPOS reports are categorized from a logical point of view which allows the division into the following four classes:

1. Input for the simulation

- a) *Volume plans.* The volume plans are the basic information for every planner within the company and the input for the simulation.

- b) *Simulation model.* As the simulation model is automatically generated from the database some reports are needed that show what the models stored look like. Thus their characteristics and consistency can be examined.
- c) *Consistency checks prior to simulation.* In order not to waste any time some checks assuring consistency of the model can be carried out prior to the simulation.

2. Monitoring the process of model generation

- a) *Server state.* With the model generator running on a Linux or AIX machine, it is difficult to watch the state of the generator. As the generator writes its state regularly into the database a web report can show a history of generator states.
- b) *Simulation runs.* As the model generator processes scheduled simulation requests a survey on the finished requests is necessary.
- c) *Current simulation.* In order to monitor the current simulation run it must be possible to observe the log file of the model generator.

3. Simulation results

- a) *Detailed result charts.* These charts are designed for the simulation expert to debug the simulation results. All the results of a simulation run are accessible.
- b) *Commented standard report.* This is a manually created report based on the detailed simulation results. It is used on the management level and focuses on the main trends and problems revealed by the simulation.
- c) *Special charts supporting the main business processes.* These special charts are directly accessible via a link and offer special information. They are supposed to replace charts created manually by planners, for example build programs over time, bottleneck charts (Fig. 8.4), or capacity charts.

4. Plan/Actual reports

- a) *Comparison of simulation parameters.* Some parameters of a simulation model are entered manually as plan parameters. If it is possible to compare these plan parameters with actual values, the parameters of the simulation model can be validated. This leads towards the validation methods of the whole simulation model.
- b) *Comparison of simulation results.* The calculated performance measures can be compared to their corresponding performance measures from actual data systems. This leads towards logistical quality control charts as presented in chapter 11.

8.3. Input for the Simulation

EPOS does not offer a graphical presentation of the complete simulation model prior to the simulation. But the reports in this section allow deep insight into the structure of the simulation model.

8.3.1. Volume Plans

The business process described in section 2.9 starts with the import of the volume plan into the central plan database. This data is needed as the demand for the simulation and its visualization gives the first impression of what results are to be expected. Moreover, the volume plan is important for other planners as well.

The volume plans can be accessed via the web browser as shown in figure 8.2. The left frame allows to select the volume plan version. The versions shown are categorized according to the location, production line, and the type (build or ship) of the volume plan. Two different views on the volume plan are provided: a stacked area chart and a table. Each week in these reports corresponds to a scenario that is simulated.

8.3.2. Simulation Model

There are five reports that show the structure of the simulation model and try to reveal inconsistencies.

The reports *Additional process steps* and *Missing engineering input* focus on the inconsistencies between the process step definitions and the process flow. The process steps are defined by the engineers whereas the product flow gets imported from the shop-floor-control system. As the process steps

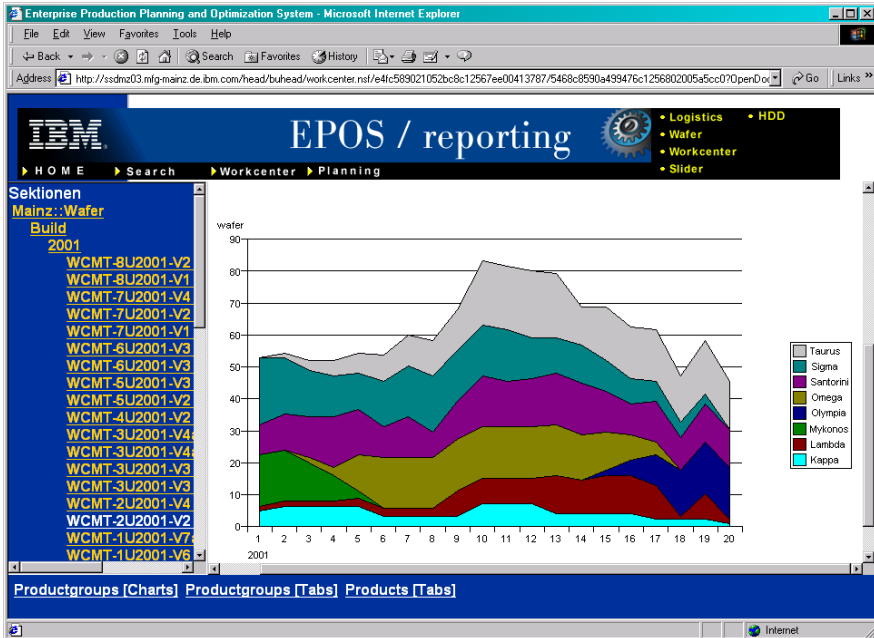


Figure 8.2.: Volume plan report

in EPOS are defined independently of the process flow it is possible that both are inconsistent in two different ways:

1. Process steps that are defined by the engineers are not part of the process flow. These are called additional process steps and are simply ignored in the model generation process.
2. Process steps that are not defined by the engineers but are part of the imported process flow are called missing process steps. These influence the simulation results and thus the administrator of the system should take care that this report does not show any entries.

Both reports simply list the superfluous or the missing process step definitions, respectively.

The report *Simulation routings* lists the main process flow and the rework

flows (see section 4.3.5) for each product in a table. The operations are sorted according to the main flow or according to the routing type. Thus it is possible to get an overview over the rework involved in the simulation studies.

The report *Junctions in the flow* shows process steps with identical operations but different work centers. This means that the operators can choose one of the two (or even more) work centers. The influence of such junctions in the process flow on the simulation results is examined in detail in section 10.

The report *Scrap* shows a list of the operations in the process flow at which scrap occurs. A more sophisticated view on the scrap along the main process flow is realized by the interactive report presented in figure 8.3.

8.3.3. Consistency Checks Prior to Simulation

Three reports can be used prior to the simulation to ensure that the simulation will be successful:

- Simulation check by volume plan version
- Yield by product group and comment
- Yield overview

The report *Simulation check by volume plan version* contains a number of heuristic checks. These should be considered prior to starting a new simulation run as many problems that might occur during the automatic model generation can be figured out in advance. As the business process shown in figure 2.10 is based on the volume plan, this report takes a volume plan version as its parameter that can be selected by the user.

The first test is to check whether the volume plan version entered by the user is the same as the one specified as the current volume plan version for the production line. As a simulation request might be defined to take the current volume plan version, this test issues a warning if the selected version is not identical to the current version of the production line. Thus the user gets reminded to adjust the volume plan version in the production line and does not start a simulation for a wrong volume plan.

Next, the user gets a survey of the mapping from products (as specified in the volume plan) to product groups (as used in the simulation). This allows to check that the correct products and product groups are simulated.

The third check shows the product groups for the simulation, their current flags, their average weekly going rates (over all weeks of the build program), their numbers of main routings, their numbers of sector rework routings, and their numbers of scrap records. This data reveals whether routings are missing and whether all products will be simulated (only those that are current).

The next table in the report shows all sources of the process flow for each product group. The user has to check that each product group has only one source operation. This operation is the beginning of the process flow.

By analogy, the next table contains all sink operations. These are operations which do not have a successor. There should only be one sink operation per product group. The sink operation is the last operation of the process flow.

To ensure a proper process flow the next table shows operations in the main flow that have more than one successor. The sum of the routing probabilities and the number of routings leaving a specific operation is shown for all operations for which the sum of the probabilities and the number of successors is greater than 1.0. This table has to be empty in order to enable a successful simulation.

The next table contains hints on possible cycles in the process flow. Three operations (op1, op2, and op3) which are connected by two routing entries with the corresponding sequence numbers are shown, if the second sequence number is smaller than the first one. Normally, the flow can be sorted by the sequence number. Thus the rows of the table are hints for cycles in the process flow, though not all hints actually have to be cycles, i. e. this check is a heuristic one.

The last consistency check shows unused scrap operations. The administration of process flows allows to copy scrap records from one product group to another. As the operations at which scrap occurs for one product are not necessarily part of the flow for other products, some scrap records might not be connected to the flow. These are ignored in the simulation and are therefore shown in this table.

The report *Yield by product group and comment* allows to show the yield loss along the main flow in the production line. The yield for the operations

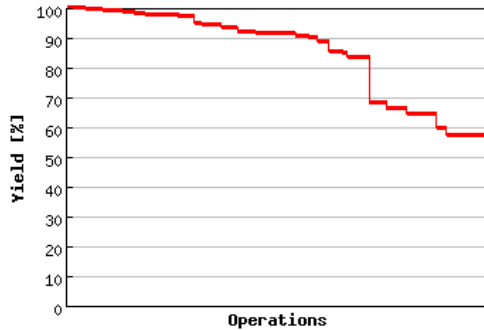


Figure 8.3.: Interactive report: Yield profile

is multiplied along the main process flow. This is an approximation as it neglects the correct process flow and the influence of rework. If the operations are sorted according to the process flow the yield loss along the line can be visualized as shown in figure 8.3. The interactive report *Yield overview* allows to get the yield charts for all product groups at once.

8.4. Monitoring the Process of Model Generation

The three reports described in this section allow to monitor the model generator:

- The *Server state* report shows the states the model generator was in at different time-stamps.
- The *Simulation run* report shows which runs are stored in the database.
- The *Current simulation* report shows the output of the model generator's current run.

The *Server state* report shows a table with the four columns *Timestamp*, *Server No.*, *State*, and *Comment*. The column *State* may contain one of the following entries:

- *Alive*. The model generator is idle and waits for requests.

- *Start-up*. The model generator is started.
- *Server-info*. After the start-up the model generator shows information on the system it is running on.
- *Simulation-run*. Shows when a new simulation-run is started.
- *Simulation-run finished*. Shows when a simulation run is finished.

All state descriptions have an associated time-stamp and comment. The server number allows to distinguish between different restarts of the server.

The report *Simulation runs* provides a list of all simulation runs stored in the database. For each simulation run the following information is provided: the simulation number, the name of the simulation run, the simulated production line, the state of the run, whether the run should be used for reporting, the person who requested the run, the time-stamp, an error, and the production line number. This report gives an overview of different simulation runs and allows to monitor the automatic simulation, to reveal errors, and to look for special simulation runs of different simulation studies if the results are made persistent in the database.

The report *Current simulation* shows the progress of a currently active simulation run. This is the same information as in the server's log file. If there are no current simulations the table is empty. Otherwise it shows the current state of the simulation together with the simulation number, a comment, and a time-stamp. This report allows to monitor the current simulation run without the need to access the simulation server directly via the telnet program.

8.5. Simulation Results

This section presents different methods how simulation results are distributed within EPOS:

- a detailed report of all results, including tables and charts,
- a commented standard report for management presentation, and
- direct access to selected results from the Notes database.

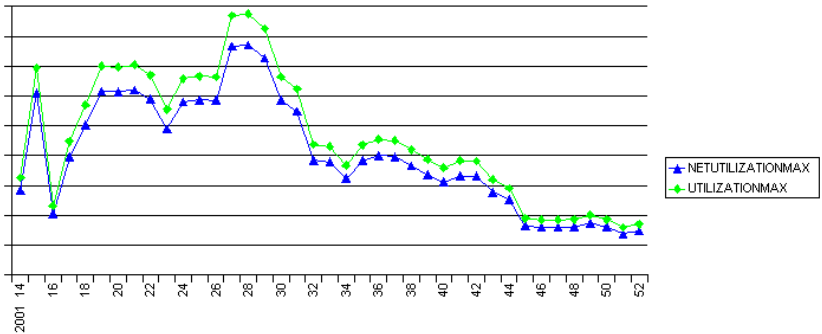


Figure 8.4.: Overall results for production lines

8.5.1. Detailed Standard Simulation Report

The standard detailed simulation report allows a hierarchical view on the simulation results. See figure 8.4 for an example. The report is divided into several sub-reports that can be accessed via links in the bottom frame. The frame to the left allows to select different categories for each sub-report. The categories are dependent on the sub-report selected. This report is a static report, i. e. static HTML pages are generated regularly by a scheduled process.

The first sub-report *Simulation* contains administrative information of the simulation run, i. e. the simulation number, the creator of the run, the time of simulation, the volume plan version etc. There are no categories to choose in the left frame.

The second sub-report is called *Line*. The charts belonging to this report show the overall performance measures of the production line over the time axis of the build program:

1. Work-in-process and lead time
2. Maximum total and net utilization
3. Daily going rate and maximum daily going rate
4. Raw process cycle time

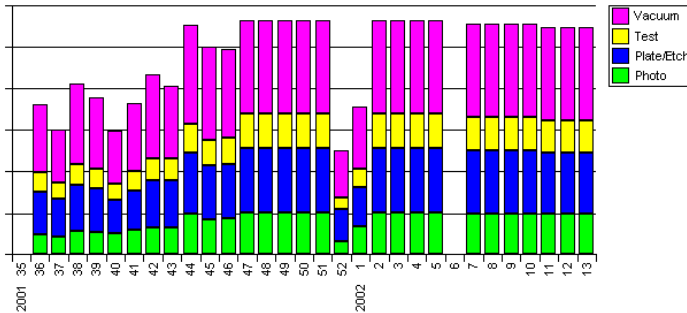


Figure 8.5.: Work-in-process by cells

5. Yield

This sub-report does not offer any categories, either. If the simulated production line is overloaded, i. e. the utilization of the bottleneck exceeds 100%, then the performance measures apart from the utilization cannot be calculated. In this case the corresponding charts have gaps.

The third sub-report *Cells (WIP)* shows the work-in-process by the cells of the production line over the time axis. There are no categories to choose in the left frame.

The fourth sub-report *Work centers* focuses on the bottlenecks of a production line. The time axis can be found in the left frame, as detailed information for each week is shown in the main frame, see figure (8.6):

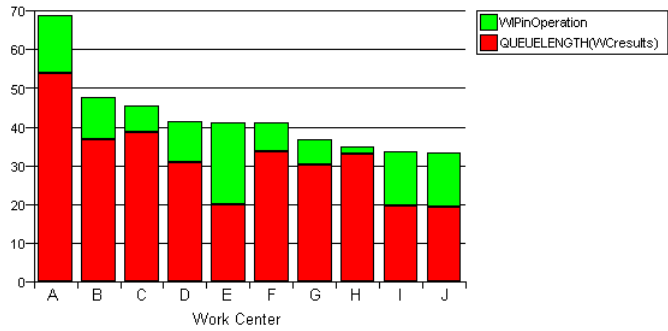
1. *Bottlenecks*. The bottleneck chart shows the total and net utilization of the 15 most utilized machines. The chart is sorted according to descending total utilization. The most utilized work center is the bottleneck (a).

Two types of utilizations can be distinguished: The total utilization is the utilization during the time the tool is ready for production, i. e. after the tools availability has been decreased by down times, breaks etc. The net utilization could be achieved if the tool's availability was 100%, i. e. if there were no breaks and down times.

The net utilization is always lower than or equal to the total utilization. If the total utilization of a work center is greater than 100% then



(a) Utilization (bottleneck chart)



(b) Work-in-process by queue length



(c) Work-in-process (WIP)

Figure 8.6.: Simulation results for work centers

the expected performance measures like work-in-process and lead time cannot be computed. If for such a work center the net utilization remains below 100%, the production process might become feasible by decreasing the down times of that work center.

2. *Top work-in-process contributors.* The total work-in-process at a work center is the queue length plus the parts in operation. Two charts show the work-in-process situations at the work centers. Each shows the expected work-in-process divided into the expected queue length (parts waiting) and the number of parts processed on the average.

The difference in the charts is the sort order: Whereas the first is sorted according to total work-in-process (the sum of the red and green bar¹ for each work center is decreasing), the second one is just sorted according to the queue length. Thus work centers can be spotted that have large queues in front of them neglecting the parts in operation.

First chart (b): Total work-in-process sorted decreasingly The summed bars are getting smaller from left to right. Each work center shows a different efficiency.

Second chart (c): Total work-in-process sorted by queue length The summed bars do not necessarily decrease from left to right, but the red bars do. Thus this chart shows the longest queues in the production line ignoring the parts in operation.

3. *Top lead time contributors.* The total lead time at a work center is the waiting time plus the service time.

As for the work-in-process charts, there are two charts showing the lead times at the work centers. Each of these shows the expected lead time divided into the expected waiting time and the service time (the time the parts are in operation).

The difference of the data represented in the charts is once again the sort order: Whereas the bars in the first chart are sorted according to total lead time (the sum of the bars for each work center is decreasing), the second one is sorted according to the waiting time. Thus, work centers can be found that have large waiting times in front of them neglecting their service time.

¹dark and light gray, respectively

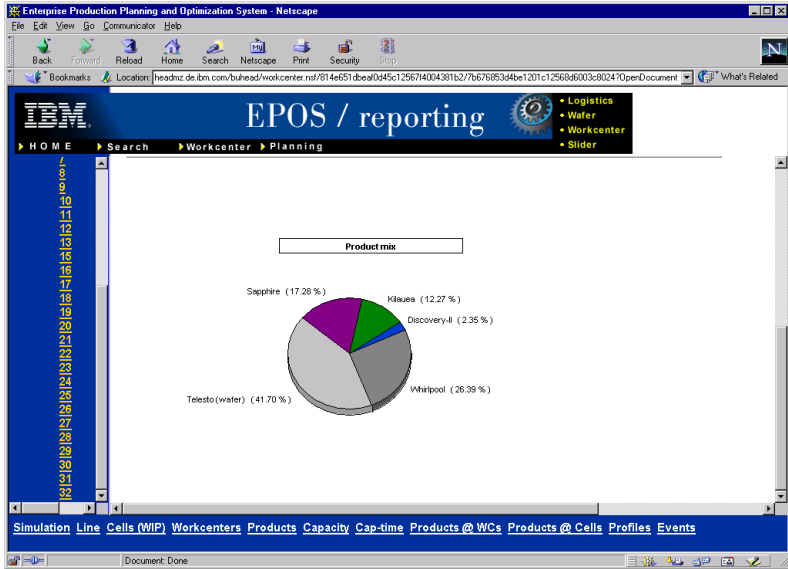


Figure 8.7.: Results for products

Note that the simulation results are given in minutes, thus the division by $1440 = 60 * 24$ (minutes per day) provides the lead time results in days.

The fifth sub-report *Products* presents a table of the performance measures of products and a pie chart for the product mix (figure 8.7). The performance measures shown are the maximum weekly going rate, work-in-process, yield, raw process time, and the overall lead time for the product.

The sixth sub-report *Capacity* shows the capacity of the production line for each week. The weeks can be selected in the left frame. A stacked bar chart shows the maximum weekly going rate based on a 6-day-week and a 7-day-week for the 20 most utilized tools. To compare the calculated capacity with the capacity needed, the quantity of products to build in the corresponding week is shown as a line. If the bars exceed the line for the build program there is enough capacity to fulfill the specified demand. Moreover, the chart shows the total utilization of the work centers. The capacity as a

weekly going rate is calculated as

$$\text{WGR}_{\text{Tool}} = \text{WGR}_{\text{max}} \cdot \frac{\text{util}_{\text{Bottleneck}}}{\text{util}_{\text{Tool}}}$$

where WGR_{max} = maximum arrival rate $\cdot 1440 \cdot 6$ or 7 for the 6-days going rate or the 7-days going rate, respectively.

The seventh sub-report *Cap-Time* shows the capacity over time for each work center. The work center can be selected in the left frame and the stacked bars show the capacity for a 6-days-week and a 7-days-week over the periods of the volume plan. As for the previous report the total number of products from the build program and the total utilization is shown. The chart shows the dependency of the capacity on the demand.

The eighth sub-report *Products@WCs* shows the performance measures of products at the work center level. A table shows the number of visits, work-in-process, lead time, and the queue length of a product at the work center. Moreover, these measures are shown as pie charts.

The ninth sub-report *Products@Cells* is similar to the one showing the work-in-process at cells (figure 8.5). It shows the lead time by cell and product for the current week only as this report is quite large and very specific. However, the data stored in the database is available for all weeks and can be recalled if necessary.

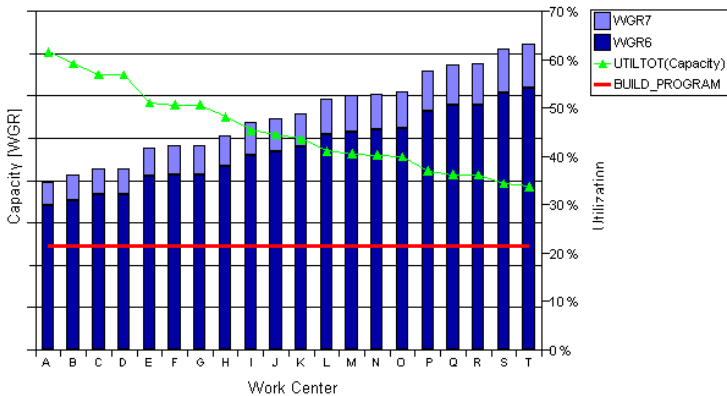


Figure 8.8.: Capacity chart

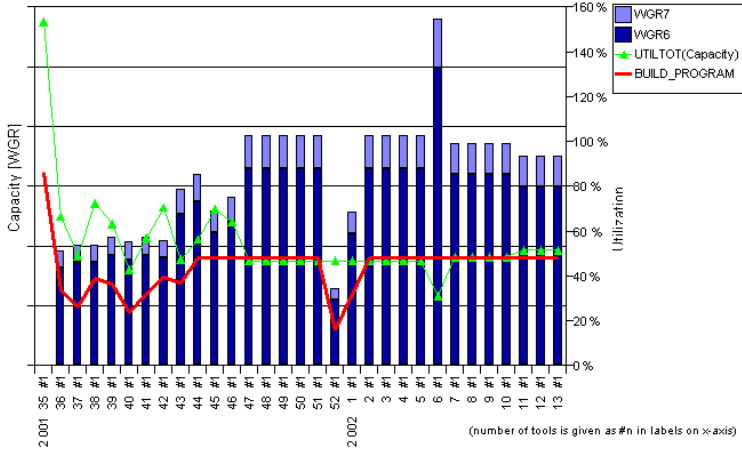


Figure 8.9.: Capacity over time

The tenth sub-report *Profiles* allows a further in depth analysis of the simulated production system. The results of several simulation runs with increasing demands are shown. Each of the three performance measures is shown over the increasing load of the system. Thus, three line profiles are constructed:

1. *Lead Time*. The lead time increases with increasing load of the system in highly loaded systems. The so-called bathtub curve which is typical of batch processing systems can be seen in the picture (see figure 8.11). The large lead times for high loads are caused by congestion of the system. For production lines with batch processing work centers, the curve increases towards lower system loads as parts have to wait in front of the work centers till their production batches are complete.
2. *Work in Process*. For highly loaded systems the work-in-process increases with increasing load. Interesting to note that the work-in-process does not increase linearly: At a certain load level the work-in-process seems to explode. This point is different for each production line and depends on all of the parameters as cycle times, routings, reliability, etc. A planner should take this point into consideration as at the same time the lead time explodes.

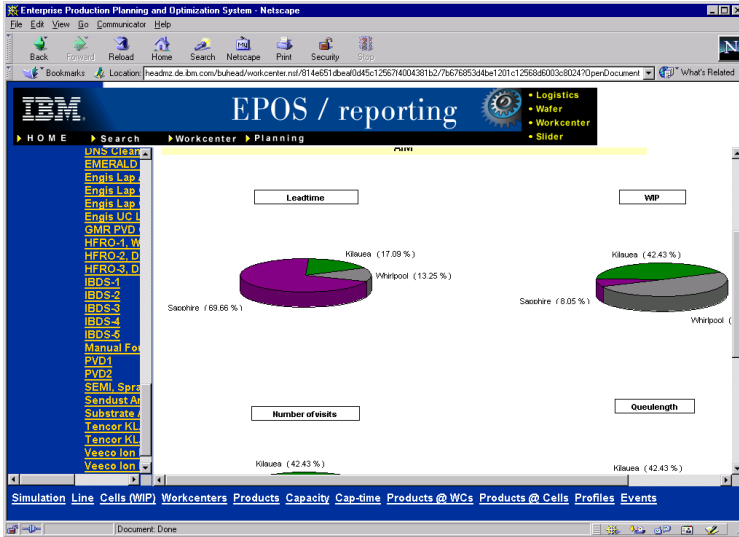


Figure 8.10.: Results for products at work centers

3. *Efficiency.* Efficiency is the ratio of overall lead time and raw process cycle time. This chart shows that the best efficiencies are achieved at a load level strictly below 100%. Lower demands lead to unused capacities and parts waiting in front of batch processors. Higher demands lead to the congestion of the system.

The eleventh sub-report *Events* shows a table of events that occurred during the model generation. The events are sorted by severity (see section 7.4.4). This report allows to find problems and inaccurate data during the automatic model generation.

8.5.2. Commented Standard Report

The goal of the commented standard report is to give a survey on the most important aspects of the simulation results. This report is to be shown monthly on the main production meeting as described in the business process in section 2.9. As the amount of result data produced for a simulation run for all weeks of a volume plan is too large to be discussed in detail, the most

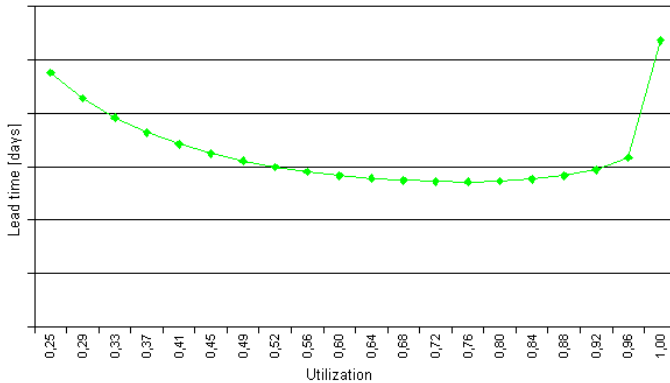


Figure 8.11.: Lead time profile

important results, problems, and trends are taken from the automatically generated reports and are enhanced with short descriptions.

As simulation results cannot be interpreted correctly without the underlying assumptions the commented standard report consists of two parts: These are, firstly, the assumptions and prerequisites for the simulation and, secondly, the selected and commented simulation results. The first part of the commented standard report contains:

- The volume plan shows which products are to be produced and their quantities (figure 8.2).
- The product mix graph shows the development of the volumes of the products (section 10.2.1).
- The report for the missing process steps shows quantity of defined process steps (section 8.3.2).
- The yield charts visualize the loss of yield due to scrap along the production line (figure 8.3).

The result section contains the charts from the report of the detailed simulation results:

- utilization over time (figure 8.4),

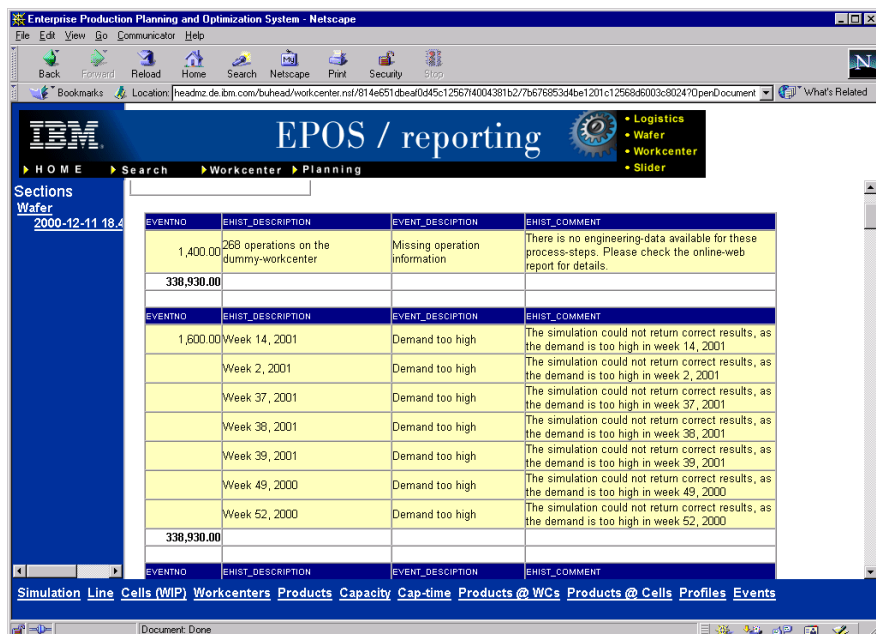


Figure 8.12.: Events during model generation

- work-in-process and lead time over time,
- bottleneck charts for selected weeks (figure 8.6),
- capacity charts for selected weeks (figure 8.8),
- capacity charts over time for selected tools (figure 8.9), and
- work-in-process line profile.

Moreover, the charts are commented. Thus planners and managers are able to get a quick overview on future performance measures. The standard report cannot be generated automatically as the planner has to select the most important charts of the simulation of a complete volume program. The charts can be copied directly from the automatically generated web site into a Notes document so that some explanatory comments can be placed next to them.

8.5.3. Accessing Simulation Results

Apart from the static reports of the simulation results described in section 8.5.1 it is also possible to access the results in an interactive way. On the one hand, this allows the integration into the system: The results for a work center can be accessed via a link from the Lotus/Notes tool-parameter-sheets. On the other hand, it is possible to browse through the simulation results while the simulation run has not finished. Both approaches are described in the following.

A tool-parameter-sheet in Lotus Notes shows all parameters entered for a work center. The integration of the simulation requires that the results are accessible from within the tool-parameter-sheet. This is achieved by a button in the sheet that opens an URL link to an interactive report. The information displayed for the currently selected work center is

- the capacity over the time axis,
- the performance measures like work-in-process, queue length, etc.,
- the performance measures for the product groups processed, and
- the performance measures for the process steps.

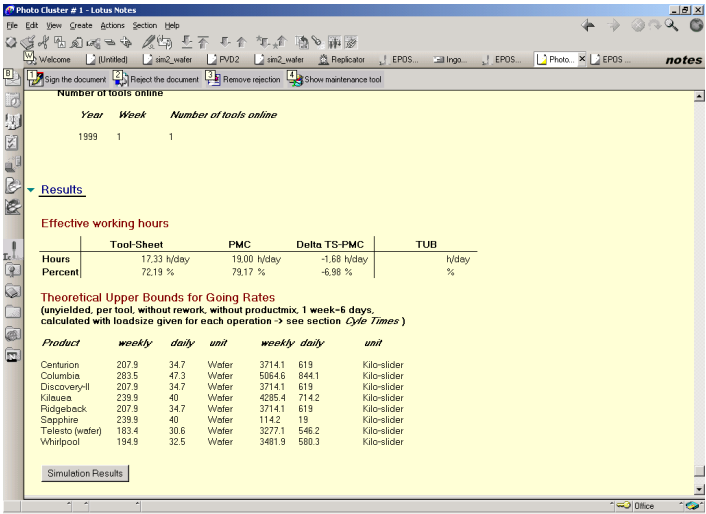
This requires that there is a topical simulation available all the time. As there might be several simulation runs for a production line, one simulation run must be chosen. The one to chose is the simulation run which is marked as the *active production simulation*. Its results are used for the static and dynamic reports.

Results that are time-dependent, i.e. that are computed for a special week can be selected by the user in the interactive report. The default is the current week. The part of the interactive report showing the work center's capacity over the time axis is shown in figure 8.13. As Lotus Notes is able to display intranet pages (rendered by the MS Internet Explorer), the user does not realize that the simulation results are read dynamically from the database and served by a web server. This is completely transparent to the user, though it ensures that always the latest simulation results are reported.

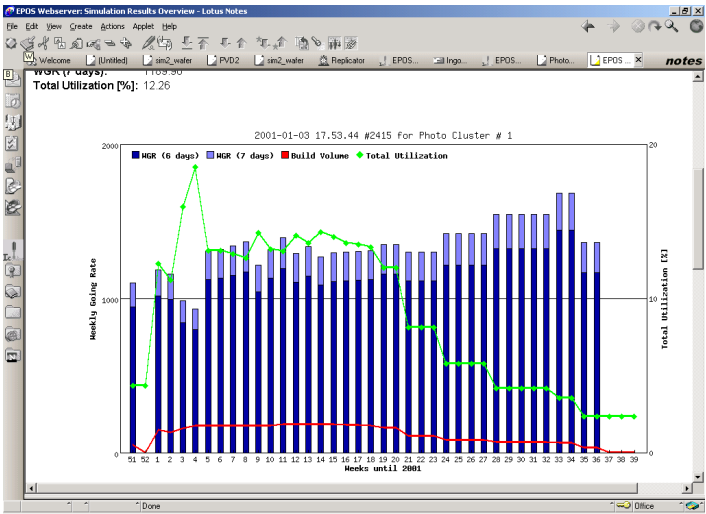
The second interactive report for the simulation results requires the user to select a simulation number. The overall performance values for the production line are shown for each simulated week. Each week provides a link

to the list of all work centers in the production line sorted according to their utilization, i. e. this is a bottleneck chart.

Making this report interactive allows to check the simulation results for a whole volume plan during the simulation, i. e. once a week has been evaluated and the results have been written into the database they can be accessed via this report. Thus it is possible to see whether the simulation run will be useful or not and the run can be stopped early, if it does not produce the expected results.



(a) Tool-Parameter-Sheet in Notes with link to simulation results



(b) Simulation result in Notes after clicking on results button

Figure 8.13.: Tool-parameter-sheets and simulation results in Lotus Notes

Chapter 9

Administration

EPOS has to be administered. The data warehouse consists of more than 100 tables in which master records, simulation models, simulation requests, etc. have to be maintained. To enable and support the administration of such a complex system graphical user interfaces (GUI) have to be developed. Different user roles, skills, environments, security considerations, etc. demand an elaborate administration architecture. However, the development of high quality graphical user interfaces (GUI) is a tedious, costly, and time-intensive task (see [PS94]). The first section of this chapter gives an overview of the requirements for the administration clients in systems for *integrated simulation*. Then different administration front-ends including the EPOS Administrator are presented. The last section of this chapter deals with the tasks to keep the system running from the technical point of view.

9.1. Requirements

A graphical user interface is the point where users interact with the system. As different as human beings and organizations are, as different are the requirements resulting from this interaction. In the following some topics which have to be taken into account when creating administration clients are discussed:

- *User roles.* Users interact with the system in different roles. Among the roles which can be found in simulation systems are capacity planners, managers, parameter suppliers from different departments (manufacturing, maintenance, engineering, staffing), simulation experts, developers, consultants, etc. (see section 2.4.1). Persons can have different roles and might even change roles in the course of time or take over other roles in different stages of a simulation project. Depending on the tasks which are associated with each role different parts of the overall information have to be administered. A good user interface ensures that a user is only confronted with the information that is required for his role. Furthermore, the possibility of editing information must be restricted according to the role that the user is assigned.
- *User skills.* It is inevitable for any real-world software system to face differently skilled users. The design of an administration front-end has to consider this. On the one hand novice users need an interface that safely directs them through the tasks they have to accomplish. The design must be focused on the ease of learning. On the other hand expert users require front-ends which support the execution of complex tasks and high transaction throughput (see [Nie01a] and [CR87]).
- *Security considerations.* Any system supporting decisions which can easily be in the range of millions of dollars needs to deploy a sophisticated security policy. By nature these policies make it hard to access the system which contradicts with the goal to make the system easy to use (see [Nie01b]).
- *Development effort.* Depending on the user's roles and skills the effort to develop administration applications can vary dramatically. *Bullet proof* applications for novice users have to be designed very thoroughly with respect to usability, learnability, and security. This leads to high development costs in contrast to just letting experts use SQL commands to change database tables.
- *Integration.* In order to gain acceptance by its users a system has to be carefully integrated into the users's IT infrastructure. This is especially important for the distributed parameter input (see section 2.4.2 and chapter 5) as the outcome of the simulation strongly depends on the correctness of the information entered. Any system that is not

integrated well into the user's environment is likely to be disapproved. Typical examples of insufficient integration of software applications are

- ▷ the requirement to use a different operating system,
- ▷ the requirement to use some software unknown to the user, e. g. a specific browser or e-mail client,
- ▷ the need to maintain data that is already available on-line in other systems, and
- ▷ enforcing the use of terminology not common in a specific environment.

Apart from the acceptance by users, integration yields other advantages like the avoidance of data conversions, the use of existing hard and software, and the knowledge and organization of established support structures. In some cases powerful software systems can be used which greatly facilitate the development process.

- *Deployment.* A corporate-wide system is likely to be installed at different production sites in heterogeneous IT landscapes. With quite frequent changes in today's IT infrastructure new versions of front-ends have to be released. For these reasons administration clients should support a fast, easy, and efficient deployment at different production sites. However, the deployment consideration depends on the number of users and their skills. If very few, skilled persons have to use an application the deployment effort becomes less important.
- *Locale-sensitive front-ends.* An administration client which is to be used enterprise-wide needs to take different user settings into consideration. This includes native language support (which might be quite necessary in countries like China, for example) and the adjustment to regional methods of entering date, time, numbers, or currency information.
- *Cost.* Like all software artifacts administration clients have to be developed, maintained, enhanced, deployed, ported, etc. These are all tasks generating costs. In addition to this some software applications often rely on other applications which require license costs. Therefore the decision which system to use requires thinking about costs at different stages of the software product life cycle (see [PS94]).

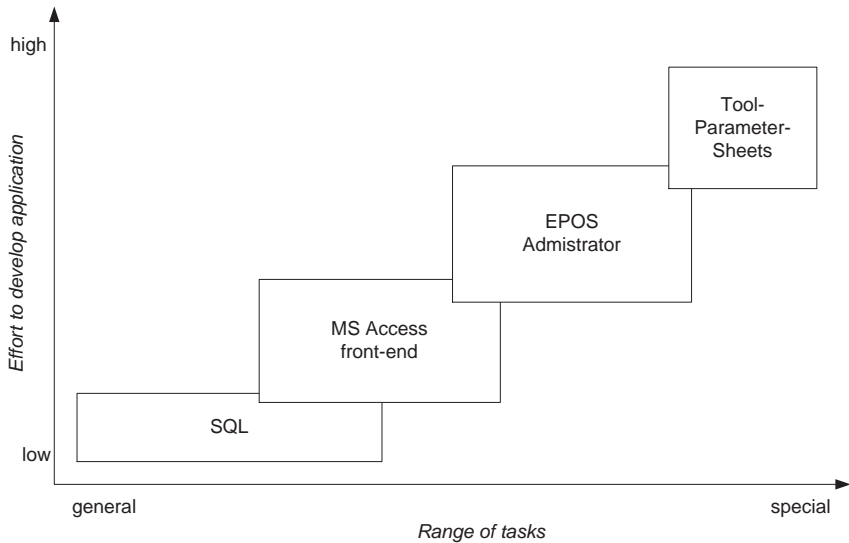


Figure 9.1.: Application to administer the simulation parameters

9.2. Levels of Administration

From the end-user's point of view a single administration client is desirable that supports all tasks, is safe and secure, and enables complex transactions and high throughput. The software architect envisions a system that is easy to enhance, maintain, and deploy. Managers and owners of the system once again have different preferences. It is evident that conflicts between these goals exist. EPOS provides four clients for administering the system instead of a single one.

Figure 9.1 shows the different applications and their position with respect to the range of tasks supported and the effort of development. This chart gives an impression of the positioning of different clients. Similar charts can be derived for security restrictions, number of users, or IT skill of users. In the following the four possibilities of administration are presented, SQL commands, a MS Access front-end, the tool-parameter-sheets, and the EPOS Administrator.

Advantages	Disadvantages
<ul style="list-style-type: none"> + Most general application: all conceivable changes are supported + Possibility of creating complex transactions using only a few SQL statements + No effort to create application 	<ul style="list-style-type: none"> - Knowledge of SQL required - Understanding of the data model including all foreign key relationships between tables required - Practically no possibility of controlling permissions on intra-table level - High risk of unintentionally performing harmful transactions - ODBC driver required

Table 9.1.: Evaluation of SQL commands

SQL Commands

The simplest way to change information in the data warehouse is to use the data manipulation language (DML) of SQL (see [HS00]). The SQL commands `insert`, `update`, and `delete` in conjunction with `select` queries are a powerful tool for the skilled administrator. To be able to use it correctly one has to understand both the SQL language constructs and the database schema of the EPOS data warehouse. These commands can only be used by the administrators of the system. Another problem is security: Production line administrators should not be allowed to use SQL statements to manipulate the data directly as access rights can only be granted on table level, i. e. if an administrator is allowed to change work center parameters, for example, he is allowed to do this for all work centers, not only for the ones in the production line that he is the administrator of. This limitation could be circumvented by the use of restricted database views for which update rights are granted to the line administrators. However, this proceeding requires to update database views, which is not always possible in current database management systems. Even if it were possible, changing user rights would ask for changes at the database schema which is not desired.

The creation of simulation models can in some cases be simplified by

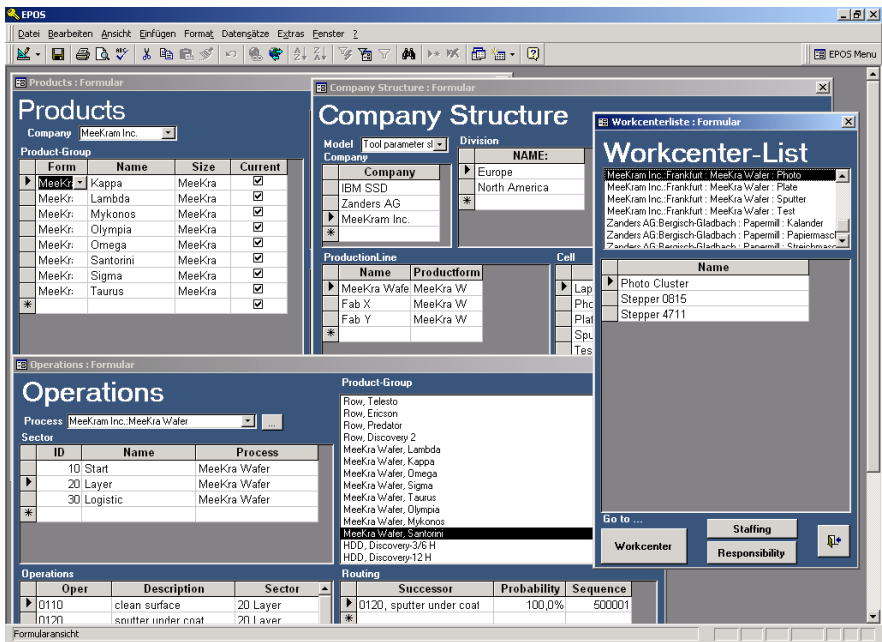


Figure 9.2.: Screen-shot of the MS Access front-end

the use of SQL statements and database imports, though. Compared to graphical user interfaces this type of model creation, offers advantages when large, simple structures are to be created. Because of the reasons stated above direct access to the database tables via SQL is only limited to few system administrators. A summary of advantages and disadvantages of this approach is shown in table 9.1.

The MS Access Front End

To facilitate the editing process for system administrators another front-end based on the desktop database system MS Access was developed. MS Access can be used as a database client without deploying its own database engine, i. e. tables from different databases (like the DB2 EPOS data warehouse) can be linked and are treated as if they were internal tables. This also allows for

Advantages	Disadvantages
<ul style="list-style-type: none"> + Fast, easy development of database front-ends + Efficient support of rapid prototyping + Possibility of updating certain database views + Joins over tables from different database management systems 	<ul style="list-style-type: none"> - ODBC driver required - MS Access license required - Database programming with Visual Basic for Applications (VBA) only - Low performance when handling large database tables - Difficult to design locale specific applications

Table 9.2.: Evaluation of the MS Access front-end

SQL views based on tables from different database management systems.

The development of database clients based on MS Access is fairly easy: First, the tables of the databases to be administered are linked via ODBC. This allows at once to use data grids to insert, update, and delete the information stored in the linked tables. Moreover, forms can be created manually or even automatically. Programs written in Visual Basic for Applications (VBA) can be used to perform more complex operations.

However, the MS Access front-end reveals the same problems that have already been discussed for SQL commands concerning access rights. It is therefore restricted to a few system administrators.

Tool-Parameter-Sheets

The opposite to SQL commands concerning security, applicability, required user skills, etc. is the tool-parameter-sheets front-end. The main users of this client are engineers entering information that cannot be taken from any other existing system. As this client has the highest number of users great care has been taken in the design phase of the graphical user interface. The access restrictions which apply are also the hardest. Grants can be given on parts of tables — either rows and specific columns. For further information about this client see chapter 5 where the tool-parameter-sheets are discussed in detail.

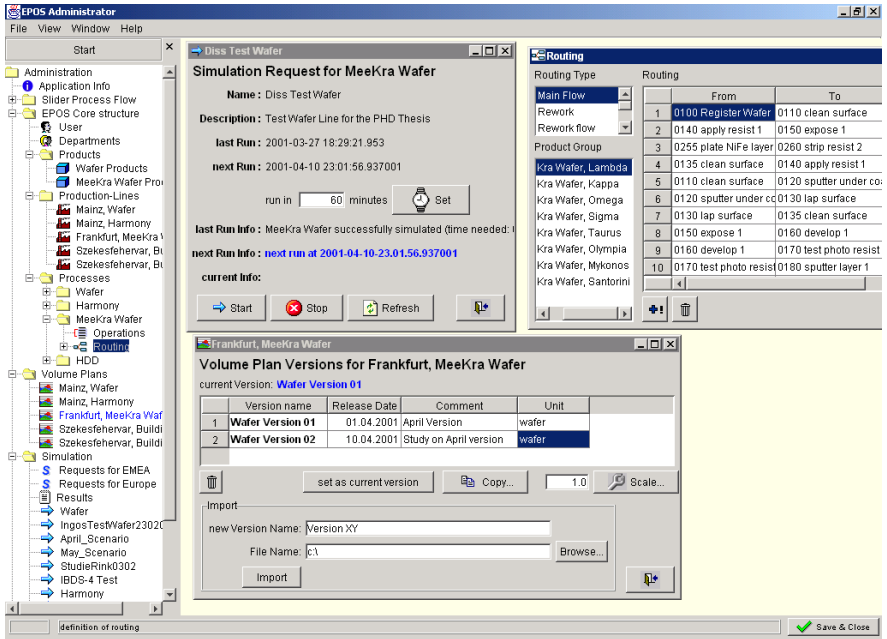


Figure 9.3.: The EPOS Administrator

9.3. The EPOS Administrator

The EPOS Administrator is the main administration front-end. Its architecture tries to circumvent the problems of SQL statements and the MS Access front-end considering the requirements discussed at the beginning of this chapter. The additional effort to develop the EPOS Administrator is necessary because of the possibility of maintaining numerous simulation models at different production sites. This feature inevitably involves several persons who are responsible for maintaining parts of the data in addition to the distributed parameter input process (see section 5.1). This requires a client capable of granting access at arbitrary granularity.

9.3.1. Overview

The users of the EPOS Administrator are production line administrators (see section 2.4.1) who need access to all parameters of their production lines. The possibility of editing data has to be limited to exactly these parameters, though. The architecture of the EPOS Administrator is consequently similar to the tool-parameter-sheets client for which the same requirements hold.

The Administrator provides a framework for the management of numerous sub-applications for different maintenance tasks. When the Administrator applet is invoked the user is already authenticated by Notes which makes an additional authentication needless. Upon startup the user is granted access to specific applications. These grants are read from the database and according to the access rights the tree allowing the navigation through the applications is constructed (see figure 9.3). To the right the desktop with some sample applications (status of simulation requests, volume plan, and routing) can be seen. Thus a user can only access and change data he is allowed to. Other information is hidden.

9.3.2. Applications

The applications provided by the EPOS Administrator reflect the tasks which have to be performed by production line administrators. These are described in the following list:

- *Users/Departments.* Editing users and departments is needed to maintain responsibilities of work center sections (see chapter 5).
- *Production lines.* A production line contains cells and these contain work centers. A production line administrator can edit all parameters of the corresponding work centers. All changes are also documented in the tool-parameter-sheets to assure consistency with that application.
- *Products.* This application enables the editing of products including the maintenance of product groups. Product forms or product types cannot be changed.
- *Processes.* A production line is assigned a set of processes which are performed in that line (see section 4.3.1). Each process consists of sectors in which operations are stored. To construct a process flow of these operations for each product group routings with the appropriate

Advantages	Disadvantages
<ul style="list-style-type: none">+ Variety of deployment options including applets+ Possibility of taking advantages of Notes security system+ use of a modern, compiled, object-oriented programming language+ Good integration into IT infrastructure+ Effortless maintenance of administrator rights in database tables+ Possibility of granting access on a arbitrary detailed level+ Possibility of creating secure, high quality front-ends	<ul style="list-style-type: none">- Medium effort to create application- Dependence on certain JDK versions- High resource demands due to Java

Table 9.3.: Advantages and disadvantages of the EPOS Administrator

probabilities can be defined. Moreover, each operation of a product group can be assigned a scrap factor which can be edited by the production line administrator.

- *Process steps.* The specification which operation is carried out for which product group on which work center is defined by process steps. This information is part of the engineering data maintained in the tool-parameter-sheets and sometimes needs to be edited by administrators, as well.
- *Volume plans.* Volume plans specifying the demand for a production line are maintained in different versions which can be copied, scaled, moved to different time frames, and aggregated over products or time frames. It is possible to change quantities and the overall plan yield specified for each product and week manually. The import of whole

volume plans is supported via CSV¹ files which can easily be created by spreadsheet applications.

- *Simulation requests.* Simulation requests are records describing a request to perform a simulation run. They contain numerous parameters which influence the scheduling, model creation, line profile calculation, etc. (see section 7.1.2). Two different applications for the maintenance of simulation requests are available:
 1. The first application makes it possible to change every parameter of a simulation request. All requests for a production line can be edited.
 2. The second application focuses on starting, scheduling, stopping, and supervising simulation runs. In contrast to the first application it is specific to a single simulation request which first has to be created by the first application. The panel shows the status of simulations currently running for this special request and the administrator gains full control over the simulation run, similar to a single user desktop system.
- *Data download.* Not all reporting tasks can be covered by standard reporting, often simulation results or input parameters are needed in applications like spreadsheets. To make EPOS data available the EPOS Administrator can be used to export the results of SQL queries against the data warehouse into CSV files². These can be opened directly by spreadsheets. The database queries can be grouped into categories and are identified by a name, the author, and the date of creation. To explain their use descriptions and comments can be entered.

To enable the export of more than one result set numerous queries separated by a semicolon can be executed at once. The resulting CSV file contains all result sets separated by blank lines and the column titles of each result set.

This feature is quite powerful as it makes any information stored in the system accessible to anybody who is granted the appropriate access

¹colon separated values

²Normally the Java security manager does not permit the access to the file system by unsigned applets. Applets in Notes are automatically signed, though, and the user is asked to grant this access or not.

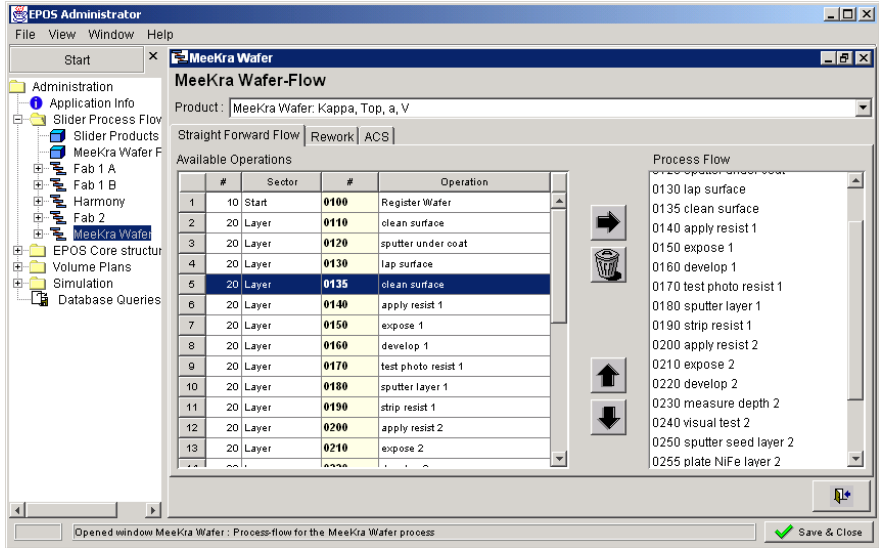


Figure 9.4.: Simplified routing administration

rights without having to install any software apart from the EPOS Administrator³.

- *Simplified routing administration.* Not all production lines use a shop-floor-control system storing a routing table that can be used in EPOS. If a process flow is maintained manually the spreadsheet-like client in the Administrator can be used to maintain it.

The manual administration of mostly linear process flows is quite cumbersome by specifying the routes, because this means to assign each operation twice, as a successor and as a predecessor. Most users are more familiar with sorted lists. The routing administration client greatly simplifies the maintenance of straight flows without sacrificing the generality of the routing graph. This is achieved by separating the administration of the main and rework flows and an additional dialog to enter scrap information: The main flow can be maintained as a

³Lotus Notes or a standard web browser supporting JDK 1.1 are required to run the EPOS Administrator.

sorted list, just the rework routings need to be entered as edges of the routing graph. Scrap can be entered per product and operation, it is subtracted later from the routing probability of the main flow during the automatic model generation (see section 7.2.5). The routing can either be maintained on the product or on the product group level. In the first case a mapping table defines how the products are aggregated to groups.

- *Interfaces.* The EPOS Administrator can be used to import data from other systems like shop-floor-control or ERP systems. The interfaces are by nature specific to the system from which the data is taken. Currently, the import of process flows from the shop-floor-control system MESA is supported. To insert process flows a mapping between EPOS product groups and MESA processes is maintained. The main process flows of this shop-floor-control system basically have the same format as the equivalents in EPOS, i. e. a successor and a predecessor operation specified by the operation ID (compare section 4.3.1), the transition probability is assumed to be equal to one. In the automatic model generation this probability is reduced by the rework and scrap rates (see section 7.2.5). These are determined by an statistical analysis of the shop-floor-control system's quality data. In the case of rework different MESA tables describing possible rework routes are joined with the rework rate. Scrap rates can be imported directly into the EPOS scrap table.

One problem during the import are operations and/or sectors which have not yet been created in the EPOS data warehouse or positive scrap and rework rates at operations which have been deleted or renamed. In these cases a dialog is opened to ask the user how to proceed (creating, skipping, etc.).

After a process flow has been imported another dialog allows to test the imported flow for syntactic correctness, i. e. existence of a single source and sink, no loops or gaps in the main flow, etc.

9.4. Technical Administration

As a complex software artifact EPOS requires the maintenance of different software components. The efforts for technical administration have to be

divided into the first time installation and the ongoing support of the system. Whereas the first one is a complex project, the ongoing support involves just monitoring a couple of servers. The following roles required for the technical support can be identified:

- *Administrator of relational database.* A database server has to be set up, must be integrated into the companies IT infrastructure, and is to be monitored and kept running. This includes the development of backup and retrieval strategies, for example. It must be assured that the database is accessible via the local area network (LAN) by ODBC and JDBC connections.
- *Administrator of Notes database.* A notes database server has to be set up (unless an existing one can be used) and the EPOS database has to be installed onto this server. If necessary, an address book containing the Notes roles for user access has to be installed as well. The Notes database contains the applets for the tool-parameters-sheets and the EPOS Administrator.
- *Administrator of simulation server.* The simulation server requires the set-up of a Linux/AIX system and the installation of the simulation software, i.e. the libraries used, the ORB, the server itself, and the model generator. All these program tasks have to be monitored and restarted in the case of failure.
- *Administration of reporting system.* Setting up the reporting system requires the installation of a Business Objects environment (server, repository, designer) for static web reports. Moreover, a Linux/PHP/-Apache system is required for the dynamic web reports.
- *Administration of interactive simulation client.* Installing the EPOS Analyzer requires to install a Java run-time environment for the MS - Windows or Linux platform and the installation of the Java program for the EPOS Analyzer.

Part III

Advanced Planning and Optimization

Chapter 10

Optimization

10.1. Problems and Methods

This chapter introduces optimization strategies to the EPOS system. Different planning tasks are identified that arise either during the model generation or production planning and the appropriate solutions are developed.

10.1.1. Introduction

The previous chapters presented the foundation of the EPOS system. Starting with an analysis of the planning problems, a data model was developed that is realized within the EPOS data warehouse. It holds a consistent set of planning parameters that serves as a basis for queueing network analysis. Looking at operations research and production management literature, like [Zäp82, Tem95, Gün93, GT00], reveals that many other planning tasks of production management rely on the same or similar data, namely process flows, cycle times, etc. This offers the opportunity of integrating other planning methods from operations research into the system.

Previously, emphasis was put on the plan parameter database, its administration, queueing network generation and analysis, and reporting facilities. This chapter takes the next step and examines how EPOS can be extended from a simulation to an optimization environment. Simulation and performance analysis are only able to analyze given scenarios but do not suggest

improvements. It is up to the user to find more promising scenarios. But most of the time the set of all parameters to change is very large, i. e. examining the effect of all possible parameter configurations (the search space) takes too long. The goal is to make use of the EPOS data warehouse and the queueing models of the production lines for further analysis that goes beyond the performance analysis by introducing methods of finding optimal or at least improved scenarios by mathematical programming approaches and by heuristic search guided by queueing network analysis.

The optimization tasks presented in this chapter arise naturally from the questions that production planners have to answer. These questions include:

- What is the optimal product mix with respect to the profit margin or with respect to lead time?
- How can process steps be assigned to a set of similar machines to achieve an equal utilization at all of the work centers or to achieve a minimum lead time?
- Which tools have to be bought to get the lead time within certain limits?

This list of typical questions from production planning and line control is by far not complete but it introduces the questions answered in this chapter. For each question the appropriate optimization method is presented, the implementation within the EPOS system gets explained, and some studies show the suitability of the methods.

The main difference between the optimization methods applied is the direct approach by mathematical programming in contrast to heuristic approaches by evolutionary algorithms. Application of the direct methods allows to get faster to optimal solutions but the problems that can be solved are limited. The power of the EPOS model is that both approaches can be combined due to the open nature of the system.

This section presents the optimization methods used whereas in the following sections the problems are examined in detail, product mix optimization in section 10.2, routing optimization/load balancing in section 10.3, and general optimization scenarios including product mix, routing, and investment decisions in section 10.4.

In general, an optimization problem includes decision variables, an ob-

jective function, and constraints,

$$\min z(x) \tag{10.1a}$$

$$C(x) \in Q \tag{10.1b}$$

$$x \in U. \tag{10.1c}$$

The *objective function* (10.1a) $z : \mathbb{R}^n \rightarrow \mathbb{R}$ describes the goal of the optimization. Usually, this is either to minimize or to maximize $z(x)$. Certain problems can be formulated with two or even more objectives, $z : \mathbb{R}^n \rightarrow \mathbb{R}^m$, where m is the number of objectives. This leads to the area of multi-criteria optimization, which is not covered in this thesis. Further references on this topic can be found in [FF93, DL94, SD95, Zit99].

The *decision variables* are the parameters that can be changed within their domains during the optimization process. One set of parameters is referred to as a *scenario* or a *solution*. Whereas simulation methods analyze one scenario and determine the corresponding performance measures, optimization methods find the scenarios that yield the best performance values or at least good approximations.

The decision variables in the case of manufacturing systems may vary widely and are strongly dependent on the time horizon (see section 1.3.1): In general, the longer the time horizon the more time there is to influence the system. For example, on the operational level the release rate, batching decisions and scheduling questions arise. The process flow is mostly fixed due to technological constraints. Thus from a scheduling point of view semiconductor manufacturing is a flow-shop scheduling problem [BESW93]. But in contrast to traditional scheduling problems scheduling such production lines is a dynamic problem in a random environment with up to several thousands jobs.

On the tactical level the decision variables are the production volumes, the product mix, and the tooling. Long-term decisions usually involve expensive investments (tools, buildings). In [HF99] the authors suggest to examine the following scenarios for short and mid term planning:

- product mix analysis,
- start rate increase,
- equipment install prioritization,
- optimal waiting time for batching scenarios,

- automatic material handling system issues,
- trade-off: capacity costs (tools etc.) vs. waiting costs (work-in-process, etc).

Within the EPOS system, the large number of parameters to modify includes integer parameters like the number of tools at work centers and continuous parameters like the product mix and routing probabilities. In general, all parameters of the simulation model might be considered as decision variables as long as the resulting scenario satisfies the technological constraints of the manufacturing process.

The *constraints* (10.1b) arise from many sources: technological constraints (for example precedence relations among the process steps), capacity constraints, financial constraints (budget limitations), etc. If a solution satisfies the constraints it is called *feasible*. It is called *optimal* if it is feasible and if it yields the largest or smallest objective function value for a maximization or a minimization problem, respectively.

Equations (10.1c) define simple constraints on the parameter domains. Normally, these are non-negativity constraints. The set of all vectors x that are considered for the optimization process is called the *search space* S . This can either be \mathbb{R}^n or U itself, if its elements can be identified efficiently.

Optimization techniques can be divided into manual search, exact methods, and heuristic search. A good overview and a discussion on heuristic methods can be found in [MF00].

Manual search is not really an optimization technique, although some authors call it that way. Manual search means analyzing different scenarios that are generated by hand. Within the EPOS system this can be done with the help of the interactive client EPOS Analyzer (see section 6.2). If a planner wants to find improvements of a production process with the help of the queueing network analysis, he can change certain parameters, start a new analysis, and compare the results from different runs. But usually the number of parameters and the ranges of their values are too large so that this procedure can only be successful in a rather limited setting.

Exact methods rely on an enumeration of the search space. Exhaustive search (complete enumeration) generates and evaluates all possible solutions and selects the best one. More sophisticated methods like branch and bound, the A^* algorithm [MF00], and linear programming [NW88, Sch98] focus on the most promising regions ignoring areas that cannot contain extrema. Other methods rely on dividing a large problem into several smaller ones

that are easier to solve and whose solutions can be combined to a solution to the overall problem (divide and conquer, dynamic programming). Unfortunately, these methods are often computationally too expensive to produce results within a sensible amount of time.

Heuristic search involves finding (near) optimal scenarios by generating promising scenarios that are evaluated by the objective function. In dependence the objective function values and the goal of the optimization further scenarios are generated. The whole process is iterated several times. The best known algorithms in this class are hill climbing and greedy algorithms. But these methods are likely to get stuck in local optima. Thus more elaborate techniques have been developed to escape those local optima. This is done by taking non-optimum solutions into consideration as well or by memorizing areas of the search space that have already been searched. Techniques following this strategy are simulated annealing [Egl90, AvL85], evolutionary computation [Gol89, SHF94, Mic96], and tabu search [Glo77, GTd93]. These techniques have been applied successfully to a variety of real-world applications.

Heuristic search algorithms require a fast evaluation of the objective function since a very large number of scenarios has to be taken into consideration. This strengthens the need for a fast computation of the performance measures by queueing theory formulae as it is implemented in the simulation server of EPOS (see section 2.5). Using this combination makes it possible to determine near optimal tool allocations, product mixes, and costs even for joint productions with random disturbances of the production process. If the calculation of the performance measures takes too long, the production will have proceeded without the simulation results having been able to take effect.

Other authors [Haj00, GP00, KW00, KB00, AFF00] try to use discrete event simulation for the performance evaluation. But for large models this approach is likely to require too much time as for a statistically proper analysis either long runs or repeated experiments followed by a comparison of multiple means (see [Har93]) have to be carried out. For a discrete simulation model of the size of the wafer lines this is hardly done within hours, which is too long for a heuristic search. A description of the problems arising in the statistical analysis of discrete event simulations and experimental design can be found in [LK91], the problems arising in the optimization of stochastic models are described in [Pfl96].

10.1.2. Linear Programming

The linear programming problem is a special case of (10.1) and is defined as follows: Find the extremum (minimum or maximum) of the linear objective function $z(x)$ with respect to some constraints on the variables and some additional linear constraints:

$$\max z(x) = c^T x \quad (10.2a)$$

$$\text{s. to } Ax = b \quad (10.2b)$$

$$x \geq 0 \quad (10.2c)$$

where $x \in \mathbb{R}^n$ is a column vector of the decision variables, $c \in \mathbb{R}^n$ is the column vector of the coefficients in the objective function, $A \in \mathbb{R}^{m \times n}$ is the coefficient matrix of the constraints, and $b \in \mathbb{R}^m$ is the right-hand side vector. The theory of linear programming can be found in many text books like [NW88, Sch98].

The IBM optimization subroutine library (OSL) which is used to solve the linear and quadratic programs in sections 10.2 and 10.3 is based on the following model which is equivalent to (10.2):

$$\max z(x) = c^T x \quad (10.3a)$$

$$\text{s. to } l_i^r \leq A_i \leq u_i^r \quad \text{for } i = 1, \dots, m \quad (10.3b)$$

$$l_j^c \leq x_j \leq u_j^c \quad \text{for } j = 1, \dots, n \quad (10.3c)$$

where $x \in \mathbb{R}^n$ is the column vector of the decision variables, $c \in \mathbb{R}^n$ is the column vector of the coefficients in the objective function, $A \in \mathbb{R}^{m \times n}$ is the coefficient matrix of the constraints, A_i is the i -th row of A , l_i^r is the lower bound for the row activity $A_i x$, u_i^r is the upper bound for the row activity $A_i x$, l_j^c is the lower bound for the variable x_j , u_j^c is the upper bound for the variable x_j . Inequalities (10.3b) represent the linear constraints whereas inequalities (10.3c) realize the domain constraints on the decision variables. The linear program (10.3) can be transformed into the problem (10.2) by adding slack variables to get equality constraints and by substituting negative variables.

Geometrically, the constraints represent a polyhedron (simplex). This is the feasible region within the n -dimensional space. An example of the special case of $n = 2$ can be found in figure 10.7 which shows the simplex defined by equations (10.17). An LP can either have an optimal solution, i.e. a point

that is feasible and minimizes the objective function, can be infeasible if no feasible solution exists, or can be unbounded.

The most common algorithm to tackle linear programming problems is the simplex method. It examines each vertex of the polyhedron along the edges until an optimal vertex is reached. This approach is justified as it can be shown that if an optimum exists there is an optimal solution among the vertices of the polyhedron defining the feasible region. The idea was mechanized algebraically by George Dantzig [Dan51]. Exact descriptions of the simplex algorithm and its extensions can be found in [NW88, p.30] and [Sch98, p.129].

The implementation of the library used for the linear programming is described in [IBM]. The IBM OSL offers different simplex algorithms (primal, dual, primal/dual) and interior point methods to solve linear programs. In section 10.2 a linear program is developed that allows to optimize production volumes with respect to capacity and some other planning constraints.

10.1.3. Quadratic Programming

General non-linear programming problems (NLP) with constraints consist of an arbitrary objective function $z(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ and arbitrary constraints. Non-linear programming is a difficult problem that has been studied in the scientific literature and no optimization method has given satisfactory results [MF00]. Standard approaches to non-linear programming can be found in [PSU88].

But special cases of this type of mathematical program have been handled successfully. One special case is the quadratic programming problem with linear constraints and a positive semidefinite matrix Q . A matrix $A \in \mathbb{R}^{n \times n}$ is called positive semidefinite if

$$x^T A x \geq 0 \quad \text{for all } x \in \mathbb{R} . \quad (10.4)$$

This assumption guarantees a convex programming problem. The quadratic program is defined as

$$\min z(x) = c^T x + \frac{1}{2} x^T Q x \quad (10.5a)$$

$$\text{s. to. } Ax = b \quad (10.5b)$$

$$x \geq 0 \quad (10.5c)$$

where $A \in \mathbb{R}^{m \times n}$ is the coefficient matrix of the constraints. $Q \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix, $b \in \mathbb{R}^m$ is a column vector of right-hand sides, $c \in \mathbb{R}^n$ is a column vector of coefficients of the objective function, and $x \in \mathbb{R}^n$ is a column vector of the problem variables.

The IBM optimization subroutine library [IBM] offers two different algorithms to tackle such a quadratic programming problem. Firstly, a two-stage simplex approximation algorithm and, secondly, an interior-point method which is a modified interior-point linear programming method. The simplex based algorithm implemented in the OSL is due to Dantzig [Dan63]. It is a generalization of the corresponding algorithm for LP problems. The interior point algorithm is a natural extension of the algorithm used to solve regularized LP problems.

In section 10.3 a quadratic program is formally developed that allows for load balancing among work centers that perform the same operations. The approach optimizes the distribution of routing probabilities that offer some natural degree of freedom in the automatic model generation.

10.1.4. Evolutionary Algorithms

Evolutionary algorithms (EAs) are heuristic search methods based on the Darwinian principles of evolution. The main factors are selection, recombination, and mutation. In the research of the last decades several different approaches have been developed, like evolution strategies by Rechenberg [Rec73] and Schwefel [Sch68, Sch75, Sch77], evolutionary programming by Fogel [FOW66], scatter search techniques by Glover [Glo77], genetic programming by Koza [Koz92], and genetic algorithms by Holland [Hol75, Gol89]. The latter are probably the mostly known kind of evolutionary algorithms.

Evolutionary algorithms have been applied to a variety of **NP**-hard [GJ79] problems like the optimization of mathematical functions by Hollstien [Hol71], Frantz [Fra72], DeJong [DeJ75] and communication networks by Davis and Coombs [DC87]. Design problems have been studied by Goldberg [Gol89] and Rechenberg [Rec73], references to the Traveling-Salesman-Problem (TSP) can be found in [Gol89, Mic96]. Other combinatorial problems have also been studied like partitioning problems [CSC93, Cur95], the fixed-charge transport problem [Pau96] and the group technology problem [Mee97, Mee01]. The application of EAs to VLSI Design problems is presented in [MPR95, JC92]. Detailed descriptions and further references on

non-linear programming problems and many others can be found in [SHF94] and [Mic96, p.15].

Common to all approaches is that they work on a set (population) of solutions (individuals, chromosomes) that are subject to certain transformations. In the course of simulated evolution these individuals fight for survival. Differences between the evolutionary techniques result from different data

```
EVOLUTIONARY ALGORITHM
Input:    Optimization problem
Output:  Approximate solution (best solution found)

begin
  Initialization of  $P(0)$ 
   $t := 1$ 
  while (termination criteria not true)
    Evaluation of  $P(t)$ 
    Selection of  $P(t+1)$  from  $P(t)$ 
    Recombination of some individuals from  $P(t+1)$ 
    Mutation of some individuals from  $P(t+1)$ 
     $t := t + 1$ 
  end
end
```

Figure 10.1.: General scheme of an evolutionary algorithm

realizations and parameters, like data structures for individuals, genetic operators, application probability of the operators, initialization schemes, population sizes, etc. Figure 10.1 shows the basic algorithm of all evolutionary techniques.

In the following let $P(t) = \{c_1^t, \dots, c_n^t\}$ be a population of individuals c_i^t at time t . A population at a given point in time is called a *generation*. Each individual c_i^t represents a solution to the problem at hand and is implemented as a special data structure S . EA literature often uses *individual* as a synonym for *chromosome*. This is at least sensible in the haploid case, i. e. if there is only one chromosome per cell, as the individual is constructed according to the information stored in the chromosome. In this case another synonym is *genotype*. The quality of a solution which is the measure for an individual's ability to survive is called *fitness*. The problem-dependent

fitness function $f : S \rightarrow \mathbb{R}$ assigns a fitness to each individual.

Starting with an initial population $P(0)$ at time $t = 0$ individuals are selected randomly from the current generation $P(t)$ according to their fitness. The higher an individual's fitness is the higher is its probability of selection. The selected individuals form the next generation $P(t + 1)$. The initial population $P(0)$ is either constructed at random and / or on the basis of certain heuristics. Some individuals of the new population $P(t + 1)$ are transformed by the application of the genetic crossover and mutation operators. Mutation operators m_i are unitary transformations $m_i : S \rightarrow S$, whereas crossover operators c_j are n -ary $c_j : S \times \dots \times S \rightarrow S$ operations; mostly, $n = 2$ for crossover operators.

The result of an evolutionary algorithm is the best chromosome of the last generation. If the best individual considered in the run of an algorithm is not contained in the final population due to stochastic (sampling) errors in the selection or replacement process, the solution can also be the best individual considered in any generation. This requires additional mechanisms, like storage for the best individual considered. The elitist strategy also assures that the best solution found is known at the end of the algorithm.

The general evolutionary algorithm shown in figure 10.1 remains the same for most problems to be solved. The implementation of an evolutionary algorithm requires the incorporation of problem-specific knowledge. The interfaces for this are the fitness function and — if more sophisticated data structures are used — the chromosome representation and the genetic operators. Moreover, for certain problems specific local search heuristics might be incorporated, leading to so-called hybrid evolutionary algorithms.

Section 10.4 presents an approach to combine evolutionary algorithms and queueing network analysis to improve tactical production planning.

10.1.5. The Goal

The goal of this chapter is to present optimization strategies that extend the EPOS framework constructed in the previous chapters. The base data for more sophisticated planning algorithms is available and is now to be used to answer further questions.

The following sections show how the techniques presented in this section are used within the EPOS system to answer the following questions from the area of production planning:

- *Product mix analysis.* What is the optimal product mix with respect to capacity and/or lead time?
- *Routing/load optimization.* How does a distribution of routing probabilities look like that minimizes load variation on tools or other performance criteria?
- *Optimization of queueing network performance measures.* How can the lead time be decreased when product mix, routings, and the number of machines are decision variables?
- *Investment analysis.* Which tools are needed in addition to the existing ones to reduce the lead time to a desired level?

The problems to the first two questions are formulated as mathematical programs that are integrated directly into the queueing network analyzer. The third question is answered with the help of evolutionary algorithms that use the performance measures calculated by the EPOS simulation server to guide the heuristic search.

10.2. Planning Manufacturing Output

This section describes how the standard model for product mix optimization can be extended to meet the requirements of real-life planning and how it is integrated into the EPOS system.

10.2.1. The Planning Task

The planning of manufacturing output includes the problem of determining the optimal product mix and belongs to the area of tactical production planning (section 1.3.1). The result of the planning process is a so-called *volume plan* or *volume program* that describes which products are to be produced at which quantities per period. Standard planning models basically take capacity requirements, costs, and prices into consideration. The models for volume planning can be divided into two classes:

- models for standard products
- models for customer specific products

In the case of standard products the planning process can be based on fixed and well-known product structures (BOM, bill-of-materials) and well-known technological processes. The planning process is independent of specific customer orders and can thus be based on total demand forecasts. Often fixed prices are set for standard products that become the constant parameters of the deterministic mathematical model.

In the case of customer specific orders goods are produced according to needs and requirements of the customers. Planning becomes more difficult as the exact bill-of-materials, the prices, and the capacity requirements are only known after the orders have been placed. Moreover, especially for customer specific products the prices are often subject to negotiations, thus making planning on the basis of contribution margins even more difficult.

The queueing analysis shown in section 3 determines the performance measures like work-in-process, lead time, etc. for a product mix given a priori. It does not try to optimize the mix. Typical questions that the product mix optimization can answer:

- Which products are to be produced at which quantities, with respect to expected maximum sales quantities estimated by the marketing department?
- How does the program change if the expected sales change?
- How does the program change if the production capacity is reduced by machine breakdowns, for example?
- How does the program change if the management decides to maximize turnover instead of margins?

10.2.2. Basic Model for the Optimum Product Mix

The goal of the standard model shown in equation (10.7) is to maximize the overall contribution (or profit) margin. The contribution margin c_j of product u_j is the difference between its price p_j and the variable costs k_j , i. e.

$$c_j = p_j - k_j. \quad (10.6)$$

The standard model for the product mix optimization can be found in [Zäp82]. It relies on the following assumptions:

- The production system supports the production of n different products by the use of m resources. Production and sales volumes are the same, i. e. no inventory management decisions are applied in this model.
- Each product has a maximum amount of production volume (sales restriction).
- The only interference of the products is with respect to common resources.
- The contribution margin is constant for each product.
- The capacities are constant within the planning horizon.
- No set-up or change over costs are considered.
- Joint production with varying ratios is not considered.
- The goal is to maximize the overall contribution margin.

These assumptions lead to the linear model

$$\max z(x) = \sum_{j=1}^U c_j x_j - F \quad (10.7a)$$

$$\text{s. to } \sum_{j=1}^U m_{ij} x_j \leq q_i \quad i = 1, \dots, W \quad (10.7b)$$

$$x_j \leq h_j \quad j = 1, \dots, U \quad (10.7c)$$

$$x_j \geq 0 \quad j = 1, \dots, U \quad (10.7d)$$

where $x_j, j = 1, \dots, U$, denote the decision variables, i. e. the production volumes of products $u_j, j = 1, \dots, U$, and the other parameters are constants having the following meaning:

c_j	contribution margin of a unit of product j ,
F	fixed costs in the planning period,
m_{ij}	processing time of product j on resource i ,
q_i	capacity of resource i ,
h_j	max volume to market of product j .

The inequalities (10.7b) are the capacity constraints, (10.7c) are the sales restrictions, and (10.7d) are the non-negativity constraints on the decision variables.

On the one hand this model has the advantage of being linear which allows the use of efficient methods to solve it, e. g. the simplex method or its extensions. On the other hand it has got some limitations:

- There might be other goals than the maximization of the overall contribution margin, for example a smooth utilization of capacities.
- The prices and the sales restrictions are determined a priori and are not subject of optimization.
- The contribution margins are assumed to be constant; discount systems can lead to non-linear models.
- The capacity constraints are fixed. In reality, working overtime can increase capacity at additional cost.
- A vicious circle between program planning and scheduling exists: Only a precise schedule can give exact information on cost and time as set-up times and change over costs are determined by production scheduling methods after the output program has been set up. But the prices are assumed to be known in the model of the product mix optimization.
- Inventory management is not considered. Especially, if sales have seasonal dependencies, products must be built ahead and stored until they are sold. This is considered by lot-sizing models (see [Tem95], for example, for further references).
- In reality, it is often difficult to determine the correct fixed and variable costs, and the exact sales restrictions are not known precisely.

Nevertheless, the solution to problem (10.7) can offer good hints for capacity planning. Especially for complicated production lines like the wafer fabrication it is often difficult to see the effects of product mix changes and to determine good volume plans.

10.2.3. Extension of the Model

In addition to the questions stated in section 10.2.1 two more topics arise in the production environment for which EPOS is developed.

Firstly, how can a given volume program be improved? Suppose the initial program has been set up by the marketing department according to current customer demands. Knowing that demands change rapidly, that finished products can be stored easily, and that the production facilities are rather expensive, planners should adjust the program so that the resulting level of utilization is adequate. This requires the basic model to be extended by another parameter per product, a lower bound for the volume of each product that assures that the minimum demands defined in the initial program are met.

Secondly, the discussion of performance analysis results with planners shows that their main focus lies on the total utilization of the bottleneck tools, i.e. the main criteria looked at is the maximum utilization of the production line. However, often the amount of wafer starts can be increased for products not visiting the bottleneck tools. For example, suppose the volumes of a product u_1 lead to a bottleneck utilization of 99%. Another product u_2 does not require any processing on the bottleneck tool. Thus its volume can be increased. To see these relationships in the queueing network analysis results is rather difficult, because they just show the results of a mix defined a priori. The product mix optimization shows directly the volumes that can be produced in addition to the predefined volumes with respect to the contribution margins assumed.

Moreover, planners might want to assure that certain lead time constraints can be met. This point leads to an integration of the methods for performance analysis and the optimization procedure and is described in section 10.4.7.

To fulfill the first two requirements, the model in equations (10.7) is extended and integrated into the EPOS core model. Let $\varrho_{\max}^* \in \mathbb{R}^+$ be an upper bound for the total utilization $\varrho_k^{tot}, k = 1, \dots, W$, of the work centers. This is the upper bound for the product mix optimization process. ϱ_{\max}^* should be chosen according to a line profile (see section 8.5.1, figure 8.11) of the corresponding production line: The inverse function of a line profile shows the utilization for a desired work-in-process or lead time level. But note that the line profile depends on a product mix defined a priori.

The decision variables are the production volumes of the products in

question. Let $x_i, i = 1, \dots, U$, be the demand of product u_i . A contribution margin $c_i, i = 1 \dots U$, is assigned to each product. These margins are the coefficients of the linear objective function $z(x)$

$$\max z(x) = \sum_{i=1}^U c_i x_i. \quad (10.8)$$

Note that the fixed costs F appearing in the goal in equation (10.7a) can be neglected without changing the optimal solution. This model assumes the margins to be input values, no variable or fixed costs are considered.

Next, the bounds for the decision variables x_i are constructed. As in the standard program the sales restrictions apply. Letting $h_i = \infty$ is possible as the resource constraints (10.7b) impose natural upper bounds on the decision variables (unless a resource offers unlimited capacity which is normally not the case for manufacturing systems). Depending on the planning scenario, two different lower bounds s_i can apply. If a complete product mix optimization is to be performed the usual non-negativity constraints apply:

$$s_i = 0 \leq x_i \leq h_i \text{ for all } i = 1, \dots, U. \quad (10.9)$$

Answering the question of what should be produced to use spare capacities a different lower bound is used. Taking the bill-of-materials and the yield into consideration, the yielded secondary demand x_i^γ as defined in equation (3.6) should be the lower bound. As this might be too high if the predefined demand exceeds capacity the following bounds apply

$$s_i = \min(x_i^\gamma, \lambda_l^{\max} \varrho_{\max}^*) \leq x_i \leq h_i \text{ for all } i = 1, \dots, U \quad (10.10)$$

where λ_l^{\max} denotes the maximum arrival rate for product l as defined in equation (3.37). Note that if $x_i^\gamma \geq \lambda_l^{\max} \varrho_{\max}^*$, setting the lower bound changes the product mix given initially by the minimum demands.

Now the constraint matrix M will be constructed on the basis of the EPOS data model. Each row i of M corresponds to a work center $w_i \in \mathcal{W}$, each column k to product u_k . The sum of a row is the work load imposed on the corresponding work center.

Let $\mathcal{A}_{l,k}$ denote the set of all process steps of product type l which are performed at work center k and $e_a \in \mathcal{A}_{l,k}$ the number of visits at process

step a . Then the matrix entry m_{lk} is defined as

$$m_{lk} = \sum_{a \in \mathcal{A}_{l,k}} \frac{e_a E[S_a]}{b_k} \quad (10.11)$$

where b_k is the batch size, $E[S_a]$ the expected cycle time, and e_a the number of visits of process step a . Note that the computation of e_a is part of the complete queueing network analysis. m_{lk} corresponds to the expected load product l imposes on work center k .

The rows are bounded from above by the time available for production of work center k . Thus the right-hand side vector q is defined as

$$q_k = \begin{cases} \varrho_{\max}^* \cdot r_k \cdot c_k & \text{if work center } k \text{ is a finite server system} \\ \infty & \text{else} \end{cases} \quad (10.12)$$

for $k = 1, \dots, W$ where ϱ_{\max}^* denotes the maximum utilization level defined a priori, r_k is the reliability and c_k is the number of tools of work center k .

To summarize in vector form, solving the following LP leads to the optimal product mix

$$\max z(x) = c^T x \quad (10.13a)$$

$$\text{s. to } 0 \leq Mx \leq q \quad (10.13b)$$

$$s \leq x \leq h. \quad (10.13c)$$

This model can be solved by the techniques presented in section 10.1.2.

10.2.4. Integrating the Product Mix Optimization

In order to integrate the product mix optimization into the EPOS system several steps are necessary:

- extension of core structures
- extension of simulation server
- changes in the model generator
- extension of result charts

- extension of the interactive client

The optimization procedure is integrated directly into the EPOS simulation server. The model generator and the interactive client just provide the input parameters and take the calculated optimal values for storage and display. This keeps changes to these clients at a minimum and allows to perform a product mix optimization on automatically generated models as well as on manually created ones.

Core Objects Extension

As the general shape of a line profile suggests (dramatic increase of work-in-process and lead time for utilization $\rightarrow 100\%$) it is not desirable to use 100% as an optimization goal for the product mix. The maximum utilization level is controlled by the parameter q_{\max}^* that is assigned to a production line. Thus the class **ProductionLine** (section 4.3.1) has to be extended by the attribute **MaxUtilizationLevel** that contains the value of q_{\max}^* . Moreover, the class has to provide attributes for the profit and the optimal profit.

The input parameters profit margin and the results for the optimal mix have to be inserted into the core model. Section 4.3.3 shows the attributes of the core classes **Product** and **ProductGroup**. Although the simulation server works on EPOS product groups instead of products, the additional attributes for the product mix optimization are assigned to the class **Product** as different products can be assigned different margins. Each product is assigned a **ContributionMargin** which is the product's price minus the variable cost of the product. Moreover, the attribute **OptPtMix** holds the optimal product mix and the attribute **OptDemand** the optimal demand under the assumption that the total utilization of the model is equal to q_{\max}^* .

Server Extension

As the product mix optimization within the EPOS simulation server requires input and output parameters, the interface definition of the simulation server has to be extended. The additional parameters have to be set and read and the optimal mix and demand values must be returned so that they can be stored in the database or visualized by the EPOS model generator or Analyzer, respectively. Figure 10.2 shows the necessary extensions to the server's interface. The complete definition of the interface of the simulation server is shown in appendix C. The constant q_{\max}^* defaults to 95%. As the

```
interface Model {

    long  setOptMaxUtilization(in double u);
    double getOptMaxUtilization();
    ...

    // performance measures
    double Profit();
    double OptProfit();
    ....
}

interface Product {

    // input parameters
    long setContributionMargin(in double cm);
    double getContributionMargin();
    ...
    // performance measure
    double OptPtMix();
    double OptDemand();
    ...
}
```

Figure 10.2.: Extension of interface definition of the simulation server

resulting optimal demand scales linearly, it is no problem to further increase or decrease the optimal demand.

Concerning the internal data structures of the server, there exists exactly one sink operation for each product. This operation is not assigned to any work center (note the asymmetry: a source operation is assigned to a work center, but a sink is not). As there is one constraint for each work center this sink operation has to be ignored. A new index map for operations assigned to work centers only avoids checking for the sink operations and speeds up the lookup as this index map contains *only* operations but no other objects like the overall index map of the server. This additional index map is constructed in the same way as the other, such that operations for products without any

demand are ignored completely during the construction of the constraints. The performance measures for those products (OptDemand and OptPtMix) are set to 0.

As the EPOS simulation server allows different time units for the demands and cycle times, the demand rates have to be adjusted to the unit of the cycle times prior to optimization. Once the optimal demand is calculated via the linear program of equation (10.13), the solution is scaled back to the time unit of the demand and multiplied by the yield, i. e. the optimal demand is calculated as

$$d_i^* = \frac{x_i \cdot y_i}{t_i} \quad (10.14)$$

where x_i is the solution to the linear program (10.2), y_i denotes the yield of product i , and t_i is the time conversion factor. Note that the yield per product does not change with varying demands and mixes. The optimal mix is calculated as

$$\alpha_i^* = \frac{d_i^*}{\sum d_i^*} . \quad (10.15)$$

The demand factor

$$f = \sum d_i^* \quad (10.16)$$

is the total volume that leads to the maximum utilization defined a priori (maximum utilization level) and is stored with each product so that the optimal demand can be easily computed given α_i^* and f .

Model Generation

As the product mix optimizer is part of the simulation server, the normal mapping described in section 7 applies. In addition, the contribution margins have to be aggregated. Within the EPOS core data model the contribution margins are assigned to products as products that are technologically equivalent might have different margins when sold to different customers. The aggregation chosen is to average the margins of the products belonging to a product group.

Within the process of the automatic simulation the optimal mix and demand is written back into the database and is finally displayed as a result on the web site of the detailed simulation results.

Simulation Results Web Site

The calculation of the optimal product mix is part of the standard simulation business process, i. e. that the optimal mix and demand are computed for each week (scenario) of the build program together with the other performance measures. The results are written into the EPOS data warehouse and the report generator creates two types of charts: a bar chart showing the current demand vs. the optimal demand and two pie charts showing the current product mix vs. the optimal product mix. A sample bar chart is shown in figure 10.3. The chart shows for each of the seven products A to G

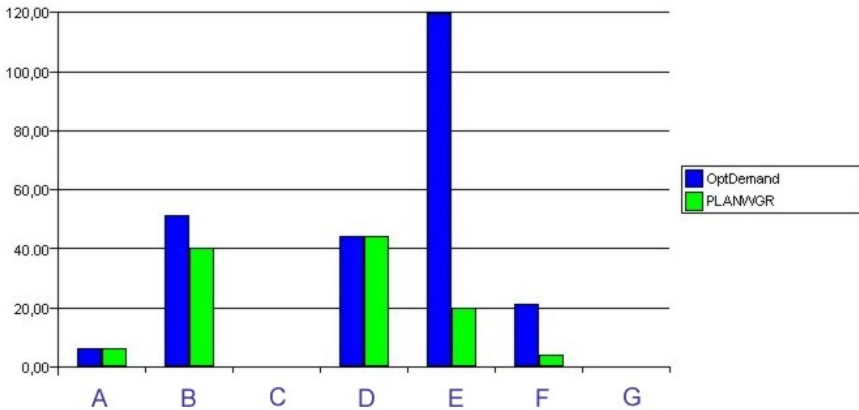


Figure 10.3.: EPOS result chart for product mix optimization

the optimum demand (left of the two bars) and the planned demand (right bar) which is the lower bound defined in equation (10.10). Products C and G do not have any demand specified, thus they are ignored by the optimizer. For products A and D just the required minimum amount should be produced whereas products B, E, and F should be produced at the increased optimal quantities shown in the graph.

Product Mix Optimization Using the EPOS Analyzer

For interactive planning the EPOS Analyzer provides the input parameters for the product mix optimization. The panel *Model* provides an input field

EPOS Analyzer

File Windows Help

Test Model

Model Workcenters Products Operations Routing Workflow Bill of Materials Resources

Add Delete Duplicate

Name	Description	Input Batch	Input Rate (SCV)	Demand	Time Unit	Contrib. Margin
P1		1	0	0	per week	0.3
P2		1	0	0	per week	0.8
P3		1	0	10	per week	0.4
P4		1	0	0	per week	0.6
P5		1	0	0	per week	0.8
P6		1	0	15	per week	0.95
P7		1	0	5	per week	0.9
P8		1	0	0	per week	0.85
P9		1	0	0	per week	0.6

Test Model

Model Workcenters Products Operations Routing Workflow Bill of Materials Resources

Model Parameters

Name: Test Model

Description:

Time Units

☐ Milliseconds
 ☐ Seconds
☒ Minutes
 ☐ Hours
☐ Days
 ☐ Weeks
☐ Months
 ☐ Years

Time Model

Hours per day: 24
 Days per week: 7
 Days per month: 30
 Days per year: 365

Optimizer

Maximum utilization level: 0.95

Figure 10.4.: EPOS Analyzer input panels for product mix optimization

for the maximum utilization level q_{\max}^* , the table in the panel *Products* is extended by an additional column for the contribution margin.

Figure 10.4 shows a screen-shot of both input panels. q_{\max}^* is set to 0.95 and there are nine products P_1, \dots, P_9 , each with a different margin shown on the right-hand side of the upper model window. The demand specified is used as the lower bound for the optimization.

Concerning the output values, the calculated performance measures are shown on two different panels as on the web site. The profit and the optimal profit are located on the panel for the overall results, whereas the optimal demand and the optimal product mix can be found in the results table for the products. Moreover, these values are shown in a bar chart comparing the current demand and the optimal demand and two pie charts visualizing the current and the optimal product mix. Figure 10.5 shows the corresponding screen-shot.

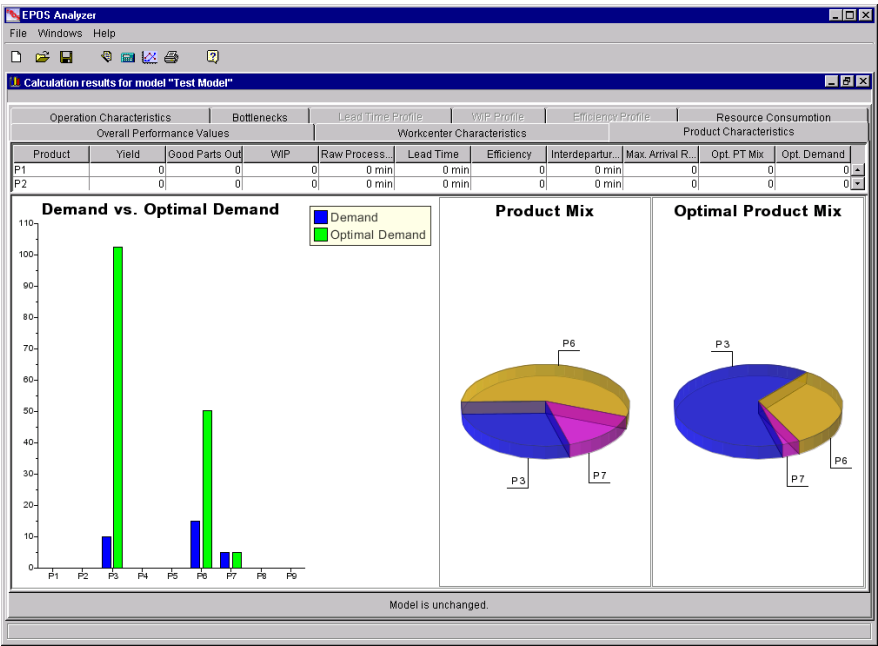


Figure 10.5.: EPOS Analyzer result panel for product mix optimization

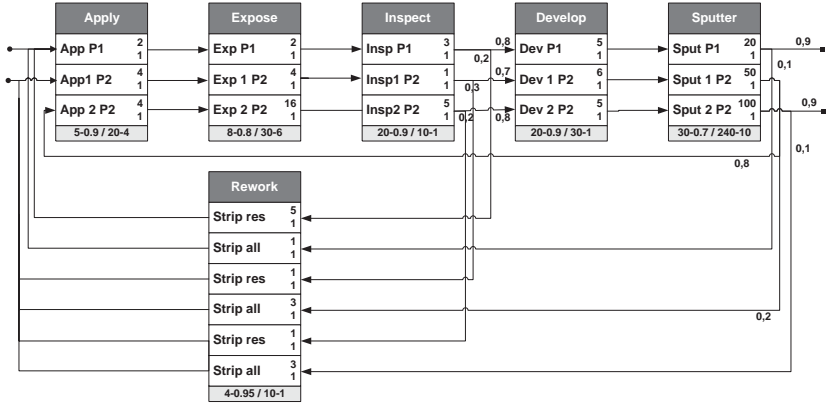


Figure 10.6.: Sample model for product mix optimization

Only P_3 , P_6 , and P_7 have a positive demand. Thus the other products are ignored and their optimum demand is set to zero. The optimum demand for P_7 stays at its lower bound defined in the input panel. The volumes for the two other products have to be increased as shown in the bar chart. The corresponding mixes are shown in the two pie charts. For further information on the EPOS analyzer refer to section 6.2 and [Reh00].

10.2.5. Example

This example is based on a small two-dimensional model. Having only two product types allows to interpret the solution graphically. The model for this example is shown in figure 10.6¹. Each constraint represents an 1-dimensional hyperplane, i. e. lines in this case. The five constraints (one for each work center) are

$$\begin{aligned}
 y_1(x) &= -0.205882x + 1.26741 && \text{Apply} \\
 y_2(x) &= -0.0921053x + 1.2096 && \text{Expose} \\
 y_3(x) &= -0.4667x + 1.9152 && \text{Inspect} \\
 y_4(x) &= -0.4x + 1.2312 && \text{Develop} \\
 y_5(x) &= -0.123077x + 1.10492 && \text{Sputter} \\
 y_6(x) &= -0.735409x + 1.76988 && \text{Rework.}
 \end{aligned} \tag{10.17}$$

¹See appendix D for an explanation of the parameters in the figure.

These are visualized in the graph figure 10.7. Depending on the ratio of the

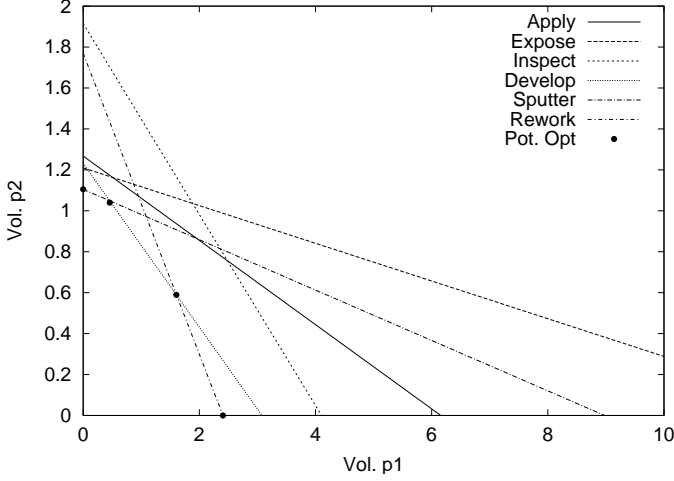


Figure 10.7.: The simplex for the sample model

contribution margins of the two products four different points (shown as + in the graph) or the connecting lines are found to be optimal. Let κ be the ratio of the margins, i. e. $\kappa = c_1/c_2$. Then the optimal solutions are:

$$\alpha^* = \begin{cases} (0, 1.10492) & -0.123077 \leq \kappa \\ (0.456, 1.04) & -0.123077 \geq \kappa \geq -0.4 \\ (1.6056, 0.5891) & -0.4 \geq \kappa \geq -0.735409 \\ (2.40666, 0) & \kappa \geq -0.735409 \end{cases} \quad (10.18)$$

Together with the vertex (0,0) these points define a two-dimensional polyhedron, the feasible region.

10.3. Routing/Load Optimization

This section describes the optimization of the distribution of routing probabilities at junctions in the process graph. From a capacity point of view, this is the question of load balancing: Suppose there is a set of work centers

performing the same operation on the same products, i.e. an arriving part can be processed on either of the work centers. Then the question arises of how to assign the routing probabilities to the corresponding process steps in the simulation model. The benefits of routing optimization are:

- *Feasibility.* If the first bottleneck can be found at the end-point of a junction, routing optimization might help to decrease its utilization. Only if it gets below 100% it is possible to determine the performance measures by queueing network analysis.
- *Finding good scenarios.* Routing optimization balances the load on the work centers at junctions and reduces the performance measures like work-in-process and lead time.

This section formally develops the optimization problem as a quadratic program and shows how to solve it with the help of the IBM optimization subroutine library.

10.3.1. Junctions

If an operation can be carried out alternatively at more than one work center, the corresponding routing probabilities have to be defined in the simulation model. These probabilities are static in the model and they are defined a priori. But nevertheless, they have an important impact on the overall performance measures of the production line as can be seen in figure 10.8. Figure (a)² shows the simple model, a network of two M/M/1/∞ stations running the same operation in parallel. There is one junction with two branches leading towards the two work centers. Figure (b) shows the lead time of good parts as a function of the routing probability x for three different arrival rates $a = 0.9$, $a = 0.99$, and $a = 1.8$. For unbalanced routing probabilities the lead time advances infinity. Due to the symmetric setting, the minimum lead time is exactly at $x = 0.5$. Note that without changing any of the work center or product parameters the lead time of good parts can vary more than 300% even in the case of moderate arrival rates. The higher the arrival rate is, the more sensitive is the lead time to the influence of the routing probabilities. This can be seen by the small (large) slope of the curves in the middle of the x-axis interval for small (large) values of the arrival rate.

²See appendix D for an explanation of the parameters in the figure.

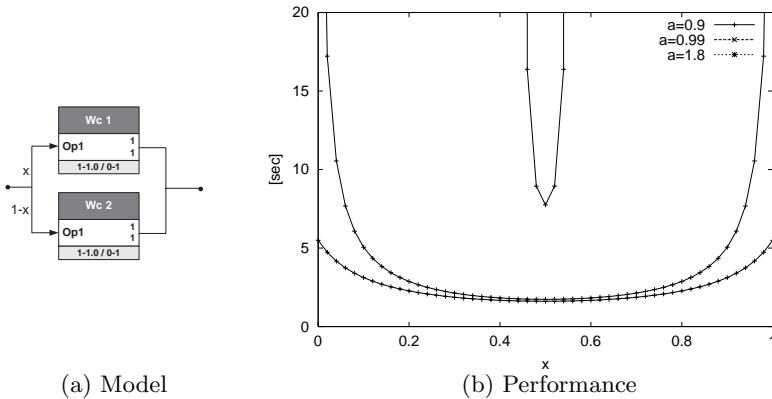


Figure 10.8.: Simple example of a junction

Additionally, the impact on the performance measures also depends on the other parameters of the simulation model like product mix, cycle times, number of servers, batch size, rework, and scrap rates etc. Inadequate routing probabilities might easily lead to wrong conclusions on the performance of the system analyzed.

Local heuristics like routing probabilities weighted according to cycle times (see equation (10.36) or section 7.2.5) face the problem of not taking the impact of rework and scrap at later stages into consideration. This is especially dangerous in re-entrant lines with many products which are typical of semiconductor manufacturing lines.

Determining an optimal distribution of routing probabilities allows to establish a simulation network that tries to minimize certain performance measures like overall lead time or work-in-process along with load differences at junctions. Whereas this section focuses on the load balance, section 10.4 takes other performance criteria into consideration. Note that all performance measures calculated by the simulation server can be used in an objective function.

The problem of determining optimal routing probabilities occurs at different occasions:

- *Automatic model generation.* There is a natural degree of freedom to specify the distribution of the routing probabilities if an operation

of the process flow is assigned to several work centers for the same product.

- *Interactive use of automatically generated models.* If a planner changes the capacity of a work center at a junction, for example by changing the number of tools, adding/removing process steps, etc., the routing probabilities have to be adjusted as well. Otherwise, the effect of the change cannot be seen. Such a change turns the probabilities determined during the automatic model creation obsolete and requires a fast re-calculation.
- *Manually created models.* In this case often very detailed information is available on scrap and rework rates for operations assigned to a work center. Complex networks with lots of parameters turn the determination of routing probabilities quite difficult.

In the case of automatically generated models there is a natural degree of freedom in specifying the routing probabilities. Figure 10.9 shows the three different representations of the routing (a) in the database, (b) in the model generator, and (c) in the simulation server³. The process flow in the database is imported from the shop-floor-control systems. It just specifies the order of the operations needed by a product ignoring the assignment to work centers. This assignment, i. e. the definition of process steps, is done separately by the engineers. Thus the database allows for the separate maintenance of process steps and the routing, which might lead to inconsistencies. These can be detected by the reports described in section 8.3.2.

The core object-model (b) combines these two data sources: The operations which are assigned to work centers by the engineers are linked according to the flow. If an operation is assigned to two different work centers, the main flow is split; this leads to a junction in the process flow. The distribution of the routing probabilities at such a junction has to be defined prior to a network analysis. This can be done either heuristically as described in section 7.2.5 or with the help of optimization techniques that try to determine a distribution that optimizes some objective. The latter approach is the subject of this and the following sections.

The simulation server provides another, similar object-model (c): Operations are assigned exactly to one work center. Thus a single operation that

³AMS stands for *Analytical Modeling System*, the CORBA module name of the simulation server.

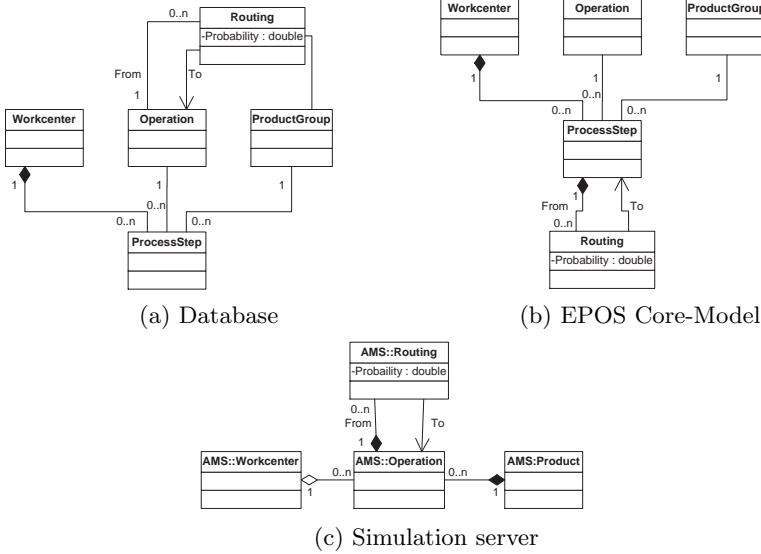


Figure 10.9.: Comparison of routing representations

is assigned to two different work centers in the core model is mapped to two separate operations in the object-model of the simulation server. In order to identify the operations in the server that originate from the same operation in the core model, a certain relation is needed in the simulation server. These ideas are formally defined in the following. Note that an operation in the object model of the simulation server corresponds to a process step in the core model.

Definition 10.3.1 (Equivalence of operations) *Let $\nu : \mathcal{A} \rightarrow C$ be a function that maps process steps into a set C of comparable elements and $a_i, a_j \in \mathcal{A}$ be two process steps. a_i and a_j are equivalent, $a_i \doteq a_j$, iff $\nu(a_i) = \nu(a_j)$.*

This defines an equivalence relation on process steps, i.e. the relation is transitive, reflexive, and symmetric.

An example of a function ν is a mapping from the process step to its product. In this case an equivalence class contains all process steps that

belong to the same product. Another possibility is to map process steps to the name of the operation, i. e. an equivalence class contains operations having the same name (and thus performing the same manufacturing step on a product). If a process step a_i is the direct predecessor of two equivalent process steps a_j and a_k , i. e. the edges $a_i \rightarrow a_j$ and $a_i \rightarrow a_k$ are part of the routing graph ($p_{ij} \neq 0 \wedge p_{ik} \neq 0$) and $\nu(a_i) \doteq \nu(a_j)$ holds, then a_i is the root of a junction in the process graph with end-points a_j and a_k .

Definition 10.3.2 (Junctions) *A junction j is a tuple (a_i, \mathcal{J}) where a_i denotes the root of the junction and \mathcal{J} the set of the process steps that are the end-points of routings leading from a_i to the equivalent process steps a_j and a_k , i. e. $a_j, a_k \in \mathcal{J} \Leftrightarrow a_j \doteq a_i$.*

Let ψ_j be the sum of routing probabilities of the edges leading from a_i to the process steps in \mathcal{J} , i. e.

$$\psi_j = \sum_{\substack{j=(a_i, \mathcal{J}) \\ a_j \in \mathcal{J}}} p_{ij}. \quad (10.19)$$

ψ_j is called junction probability. Let J be the vector of all junctions j of a simulation model, $|J| = N_J$.

Let $W_j, j \in J$, be the set of work centers of the process steps of \mathcal{J} , i. e. $W_j = \{w_k | w(a_i) = w_k, a_i \in \mathcal{J}\}$. The sets of work centers W_j compose a lattice. Let \hat{W}_j be the minimal upper bound of W_j within that lattice.

Note that $\psi_j \leq 1$ holds and that $1 - \psi_j$ is not necessarily the scrap of the junction's root a_i as there might be other routings leading to process steps a_k that are not equivalent to those in \mathcal{J} . Especially it might be the case that a_i is the root of two junctions $j_1 = (a_i, \mathcal{J}_1)$ and $j_2 = (a_i, \mathcal{J}_2)$.

As an example recall the model shown in figure 10.8. Let C be the set of all strings and $\nu : \mathcal{A} \mapsto C$ the function that maps a process step to the name of the operation. In the example $\nu(a_1) = \nu(a_2) = \mathbf{Op1}$. Then there is one junction $j = (a_{\text{source}}, \mathcal{J})$ in the model where $\mathcal{J} = \{a_1, a_2\}$. $\psi_j = 1.0$ and $W_j = \{\mathbf{Wc1}, \mathbf{Wc2}\}$, $J = \{j\}$, $N_J = |1|$, and $W_j = \hat{W}_j$. Note that in the case of a single product this definition of ν is sufficient. In the multi-product case process steps with the same name belonging to different products must not be equivalent.

10.3.2. A Quadratic Program for Routing Optimization

Routing optimization means to use the freedom inherent in the junctions of a simulation model to optimize an objective function. The freedom arises at junctions because the work performed at the end points is equivalent. The goal of routing optimization is the optimization of the performance measures of the simulation model. All routing optimizations need to fulfill the constraints of the routing graph and some additional constraints to be specified below. The basic idea is to take the system of equations for the number of visits of operations, to add some artificial variables, to turn the number of visits into decision variables and to minimize some objective function over these decision variables.

Let $\mathcal{A} = \{a_1, \dots, a_A\}$ be the set of all process steps of the queueing network, $|\mathcal{A}| = A$, and $e_i, i = 1, \dots, A$, the associated number of visits as defined in section 3.5.

Let $j = (a_i, \mathcal{J}) \in J$ be a junction. Each routing $a_i \rightarrow a_j, a_j \in \mathcal{J}$ of the junction j is assigned an artificial variable that can be interpreted as a number of visits for an auxiliary process step \tilde{a}_j inserted between a_i and a_j . Thus the model gets

$$n = \sum_{\substack{j \in J \\ j = (a_i, \mathcal{J})}} |\mathcal{J}| \quad (10.20)$$

additional artificial variables, one for each edge in a junction where J denotes the set of all junctions of a model. Figure 10.10 shows the interpretation of the artificial variables. Two artificial variables (process steps) have been inserted into the queueing network. It is not possible to use the routing

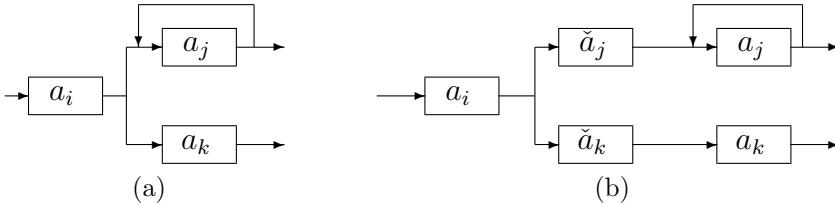


Figure 10.10.: Insertion of artificial variables

probabilities only as decision variables, as they influence the number of visits even at process steps that are not end-points of junctions. Insertion of the

artificial variables (process steps) allows to state the constraints among the number of visits of the process steps involved in a junction.

Let a_j be a process step at an end-point of a junction, e_j its associated number of visits, and \check{e}_j the associated artificial variable. Substituting $p_{ij}e_i = \check{e}_j$ in the system of linear equations for the number of visits (equation 3.7)

$$\sum_{n=1}^A (\delta_{m,n}e_m - p_{mn}e_m) = e_{0,m} \quad \text{for } m = 1, \dots, A \quad (10.21)$$

yields a modified system of linear equations for the number of visits. Note that this means that the routing probability has been replaced by the artificial number of visits following the idea of transferring decisions on routing probabilities completely to decisions on number of visits.

Next, the constraints that control the flow at junctions are needed. The junction probability ψ_j has to be constant for all different assignments of routing probabilities, i. e.

$$\sum_{\substack{j=(a_i, \mathcal{J}) \\ a_j \in \mathcal{J}}} \check{e}_j = \psi_j \quad \text{for all } j \in J. \quad (10.22)$$

This leads to one additional constraint for each of the N_J junctions.

Finally, the mathematical model of the performance analysis requires all routing probabilities to be strictly larger than 0. Consider a routing edge $a_i \rightarrow a_j$. The equations

$$e_i p_{ij} = \check{e}_j > 0 \quad \Leftrightarrow \quad \exists \epsilon > 0 : p_{ij} = \frac{\check{e}_j}{e_i} > \epsilon \quad \Leftrightarrow \quad \exists \epsilon > 0 : \check{e}_j - \epsilon e_i > 0 \quad (10.23)$$

hold. $\epsilon > 0$ denotes the minimum allowable routing probability. These constraints lead to additional n linear constraints, where n is the number of routes in junctions (10.20). Moreover, equation (10.23) shows how to compute the optimal routing probabilities from the number of visits e_i and \check{e}_j .

These constraints can be composed to form the matrix M

$$M = \begin{pmatrix} \boxed{A} & \boxed{B} \\ \boxed{C} \\ \boxed{D} \end{pmatrix}. \quad (10.24)$$

Here $A \in \mathbb{R}^{m \times m}$, with m as the number of process steps, is the modified system of routing equations $P^T - I$ of equation (10.21) and $B \in \mathbb{R}^{m \times n}$ contains the corresponding n artificial variables. Matrix $C \in \mathbb{R}^{N_J \times (m+n)}$ represents the equations for the junction probabilities of equation (10.22). Matrix $D \in \mathbb{R}^{n \times (m+n)}$ realizes the minimum bounds on the routing probabilities introduced in equation (10.23).

Given a simulation model the algorithm in figure 10.11 finds all junctions in the model if it is called with $W = \emptyset$. If only junctions at certain tools defined a priori are to be considered, then W can be initialized with the names of those tools. This is important for complexity reasons. The more junctions are to be optimized the longer it takes. But often the planner is only interested in those tools that have an important impact on the performance measures. For example, W can be initialized with the first l bottlenecks or the first l lead time contributors. The algorithm checks for all operations o_i whether it is a possible root of a junction. A necessary condition is that at least two routes are leaving o_i . But this is not sufficient as the operations the routes lead to need not necessarily be equivalent. They might lead to a main flow and a rework operation, for example. The number of equivalent operations is counted in the histogram *hist* and if entries larger than 1 are found, a new routing list and the corresponding junction are constructed.

10.3.3. Objective Function

To motivate the choice of the objective function, recall the formula for the expected queue length for batch service systems shown in equation (3.39)

$$E[\hat{Q}_k] = \frac{b_k - 1}{2} + \frac{\varrho_k}{1 - \varrho_k} P_c \frac{b_k C^2[\hat{I}_k] + C^2[\hat{C}_k]}{2} \quad \text{for } k = 1, \dots, W.$$

This formula is central to the calculation of the performance measures. In a certain scenario the batch size b_k is usually fixed as it is determined by technological constraints of the work center. Thus the factors $\varrho/(1 - \varrho)$, P_c , and the sum of the coefficients of variations have to be considered. Taking the latter for constant at the moment leaves $\varrho/(1 - \varrho)$ and P_c . As $\lim_{\varrho \rightarrow 1} \varrho/(1 - \varrho) = \infty$ and $P_c(c, \varrho) \nearrow 1$ as $\varrho \rightarrow 1$ for a constant number of servers c , ϱ is the essential variable that influences the performance measures of the queueing network. Figure 10.12 shows this influence graphically. Averaging the load of several work centers reduces at least the maximum utilization within that group of work centers. Consequently the objective described in

FIND JUNCTIONS

Input: set<string> W

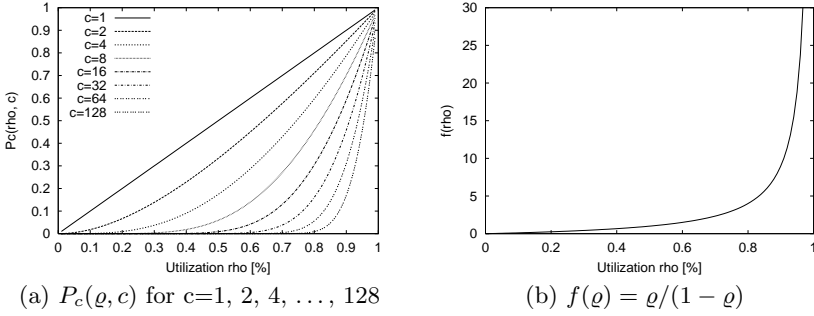
Output: list<Junctions> J

begin

```
map<string, int> hist;    // histogram
list<route> rtList;      // list of routes
list<Junctions>  $J$ ;      // list of junctions

for all products  $p_i$  {
  for all operations  $o_i$  of  $p_i$  {
    if ( number successors > 1 ) {
      // is one of the succeeding work centers in  $W$ ?
      bool wcJunct=false;
      for all routings  $r_i$  leaving  $o_i$ 
        if (  $r_i \rightarrow \text{getTo}() \rightarrow \text{getWc}() \rightarrow \text{getName}() \in W$  )
          wcJunct = true;
      if ( (wcJunct=true) or ( $W = \emptyset$ ) ) {
        // construct new histogram
        hist.clear();
        for all routings  $r_i$  leaving  $o_i$ 
          hist[ $r_i \rightarrow \text{getTo}() \rightarrow \text{getName}()$ ] += 1;
        rtList.clear();
        for all routings  $r_i$  leaving  $o_i$ 
          if ( hist[ $r_i \rightarrow \text{getTo}() \rightarrow \text{getName}()$ ] > 1 )
            rtList.append( $r_i$ );
        if ( rtList.size() > 0 ) {
           $j = \text{Junction}(o_i, \text{rtList})$ ; // constructs a new junction
           $J$ .append( $j$ ); // add junction to set of all junctions
        }
      }
    } // if more than 1 successor
  } // end for all  $o_i$ 
} // end for all  $p_i$ 
return  $J$ ;
end
```

Figure 10.11.: Algorithm to find junctions


 Figure 10.12.: Influence of ϱ on work-in-process

the this section focuses on an equal utilization of the work centers at the end-points of a junction.

Let $J = (j_1, \dots, j_{N_J})$ be the set of all junctions of a queueing model and $W_j = (w_1, \dots, w_{n_j})$ the closure of all work centers that lie at the end points of junction j . Trying to minimize the sum of mutual utilization differences leads to the following objective function:

$$\min z(e) = \sum_{j \in J} \sum_{i=1}^{n_j} \sum_{j=i+1}^{n_j} \left(\sum_{l=1}^u \sum_{\substack{p \in \mathcal{A}_{i,l} \\ w_i \in \hat{W}_j}} \frac{e_p \alpha_l E[S_p]}{b_p c_i r_i} - \sum_{l=1}^u \sum_{\substack{q \in \mathcal{A}_{j,l} \\ w_j \in \hat{W}_j}} \frac{e_q \alpha_l E[S_q]}{b_q c_j r_j} \right)^2 \quad (10.25)$$

where $\mathcal{A}_{i,l}$ denotes the set of all process steps of product l at work center i , e_p the number of visits, b_p the batch size, $E[S_p]$ the mean cycle time of process step a_p , and c_i the number of tools, r_i the reliability at work center i . α_l denotes the product mix of product l . Note that this is not directly the utilization because the product mix is included in the formula instead of the yielded secondary demand. The product mix is directly proportional to the yielded secondary demand, thus the optimum solution is the same. The coefficients of the quadratic matrix Q with $z(e) = e^T Q e$ can be computed directly. Let

$$x_{pq} = \frac{\alpha_p \alpha_q E[S_p] E[S_q]}{b_p b_q c_p c_q r_p r_q} \quad p \in \mathcal{A}_{i,l}, q \in \mathcal{A}_{j,l}; w_i, w_j \in \hat{W}_j, j \in J; l \in \mathcal{U} \quad (10.26)$$

for all process steps p, q at the work centers of the junction where c_p and r_p denote the number of tools and the reliability, respectively, of the work center the process step p is assigned to (by analogy for process step q). Then the matrix entries can be computed as

$$q_{pq} = \begin{cases} -x_{pq} & \text{if } a_p \text{ and } a_q \text{ are assigned to diff. work centers} \\ x_{pq}(w_{n_j} - 1) & \text{else} \end{cases} \quad (10.27)$$

where w_n is the number of all work centers in that junction class, and p, q as above. As $z(e) = e^T Q e$ is a sum of squares Q is positive semi-definite.

However, there is a pitfall in using this objective function. The objective function $z(e)$ in equation (10.25) includes the product mix α_l as a parameter. Note that the α_l is determined with respect to the secondary demand yielded x_l^γ which in turn is determined by the yield (see equation (3.11)). As a product's yield is the number of visits at its sink, it is as well a decision variable and thus may vary during the course of the optimization process. Thus this objective function can only be used if the overall yield of each product does not change. Otherwise, the optimal solution does not lead to the desired goal.

The substitution

$$\alpha_l = e_{\text{sink}(l)} d_l / \sum_{i=1}^U e_{\text{sink}(i)} d_i \quad (10.28)$$

could reflect the dependency on the product yield but leads to a non-quadratic goal function as $e_{\text{sink}(l)}$ itself is a decision variable. Moreover, multi-stage product structures would require to solve the system of equations (3.6) within the goal function.

Luckily, this pitfall is not that serious as it looks, because it does not apply to the models generated automatically. Section 4.3.5 shows that the routing probabilities are defined with respect to operations in the EPOS core structure. As scrap is modeled by routing probabilities, it depends on operations but not on process steps. Thus, changing the probability distribution at a junction does not change the product yield: If the same operation is placed on different tools, yielding the process steps a_i and a_j , $a_i \doteq a_j$, the probability at the junction is split, and both operations will be assigned the same scrap probabilities. The general case is more complex and will be handled in section 10.4.

To summarize, the routing optimization involves the following steps:

1. Find junctions
2. Calculate the equivalence classes and the minimum upper bounds \hat{W}_j
3. Identify artificial variables
4. Construct constraints
5. Construct the matrix of the goal function Q
6. Solve the quadratic program to yield its optimal solution
7. Calculate routing probabilities from number of visits
8. Set new optimal routing probabilities

Following this procedure the normal performance analysis of the queueing network can be started.

10.3.4. Examples

Bottleneck Charts of Wafer Line

This example is supposed to present the results of the routing optimization procedure within the automatic simulation. Figure 10.13 shows a bottleneck chart generated for a simulation of the wafer line. The first 15 bottlenecks are shown. One class of four work centers has been considered. The work centers are highlighted by the vertical boxes. The chart shows their utilizations to be 127%, 86%, 72%, and 70%, respectively. After the optimization procedure has been applied, the total utilization is 84% for each of the work centers, i.e. the assignment of process steps to the work centers allows an identical utilization for all work centers. The result is shown in figure 10.14: All four work centers are placed next to each other and fall into one large box. Note that in this case the maximum total utilization within that group dropped below 100% (from 127% to 84%). Thus the demand that looked infeasible with respect to this work center group turns out to be feasible just by changing the routing probabilities. Of course, there are two other bottlenecks that have to be eliminated first before that network can be evaluated for its performance measures by queueing theory.

The charts presented here are generated for each week (scenario) of the volume plan that is analyzed in the process of *integrated simulation*.

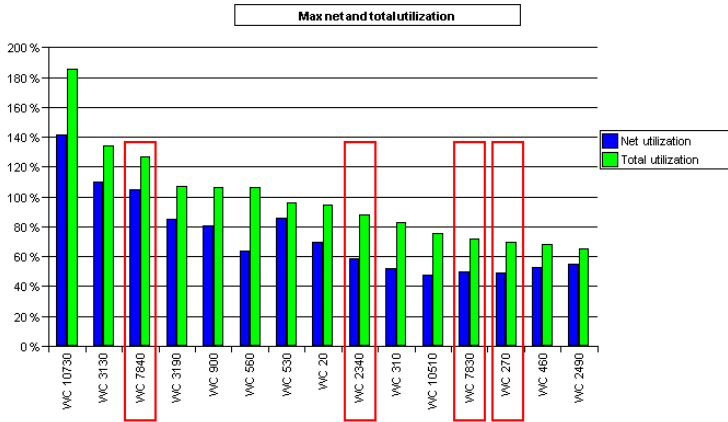


Figure 10.13.: Bottleneck hierarchy without optimized routings

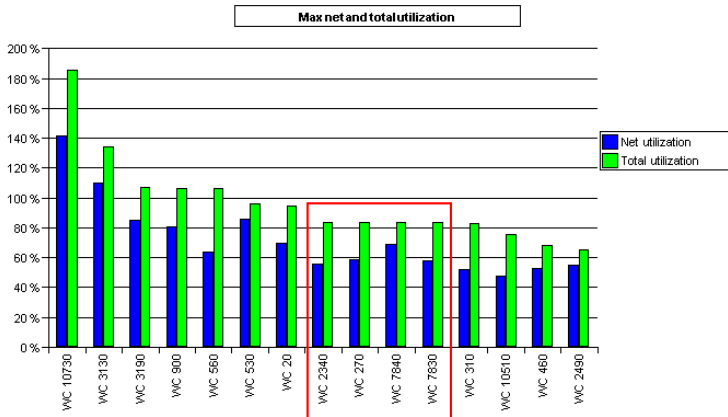


Figure 10.14.: Bottleneck hierarchy with optimized routings

Experimental Study

This example is based on a sample model whose structure and size are similar to that of a model of the wafer line. The experiment is organized as follows: The queueing model contains 27 different classes of junctions. For each class a work center is chosen that serves as a *seed*, i.e. looking up all junctions in front of that work center and constructing \hat{W}_j , the whole equivalence class is found. This gives 27 seeds. An optimization run tries to optimize a subset of all classes, given by a subset of the seeds. Let s_1, \dots, s_{27} be the seed work centers and c_1, \dots, c_{27} the corresponding classes, then optimization run i tries to optimize the junctions in classes c_1, \dots, c_i , i.e. every run includes the optimizations of the previous run, the last run is the complete optimization of the whole model. This experimental setting allows further insight into the nature of routing optimization.

Due to the construction of the optimization problem it is clear that the utilization of the work centers within a class will be as equal as possible with respect to the process steps assigned. Therefore, the question is whether this optimization leads to a better overall performance although work-in-process and lead time are not directly part of the optimization procedure. Moreover, the dimensions of the spare matrices and the number of the elements give an impression of the complexity of the problem.

The results are shown in figures 10.15(a-h). Some measures stay the same over all optimization runs. These are summarized in table 10.1. The

	3 Products	9 Products
Calculation time	6 sec	20 sec
Yield	0.51	0.57
Non-zero entries in A	8916	8916
Trigger	57.5	27.1

Table 10.1.: Parameters for sample model

calculation time of the model is dependent on the number of process steps involved, which is mainly determined by the number of products in the model that have a non-zero demand. The more process steps there are the larger the system of equations gets and the more performance measures are to be evaluated.

The model examined in this experiment is one that has been set up by

the automatic model generation. The latter have the characteristic that modification of the routing probabilities at junctions does not change the overall yield. This is why the yield remains constant over all runs. Adding products with non-zero demand leads to a different model with a different overall yield.

The number of non-zero elements in matrix A of equation (10.24) is constant as no routings are added or removed. The routing probabilities of non-zero products are represented in the matrix. They get zero coefficients in the matrix Q in the goal function.

The overall minimum and maximum utilization are only influenced if there are junctions before the most/least utilized tool in the line. Otherwise these values stay the same. That is why the performance measure *trigger*, i. e. the inverse of the maximum arrival rate, stays the same in this experiment. The most limiting work center (the first bottleneck in the hierarchy) is not part of a junction. Thus its utilization is not changed.

Figure (a) shows the number of junctions. As the seeds are chosen from the work centers at the end-points of junctions, the number of junctions increases as the number of seeds increases. Moreover, as process steps are defined on a product basis, the more products there are the more junctions are found.

Figure (b) shows the size of the sparse constraint matrix. This is given by the overall dimensions of the four submatrices A, B, C, D in equation (10.24). The curve is increasing monotonously as each run includes the classes of the previous ones. As the number of process steps assigned to the work centers may vary, the steps of the curve are not equidistant. As all process steps are considered — including those for zero demand products — the curve is not dependent on the number of non-zero products and thus is the same for the three and the nine product case.

The number of the non-zero entries in the sparse constraint matrix is shown in figure (c). Matrix B holds the artificial variables, C the constraints for the junction probabilities, and D assures that the optimal routing probabilities do no fall below a minimum value. The number of non-zero elements in matrix A is constant and given in table 10.1.

Figure (d) shows the number of non-zero entries in the matrix Q of the goal function. In the case of 9 products more process steps are to be considered, thus the number of entries is larger. Again, different junction classes may lead to a different increase in elements due to a different number of assigned process steps. The huge difference between the 3 and the 9

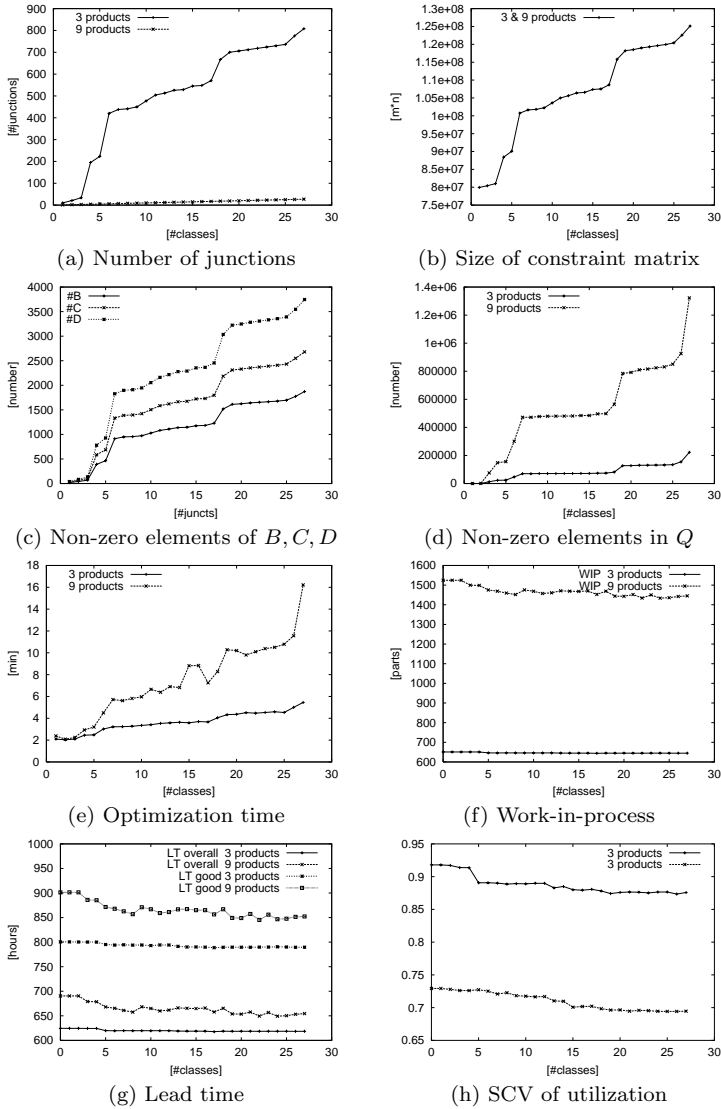


Figure 10.15.: Results of optimization experiments

products curve is due to the quadratic increase of comparisons of the process steps.

Figure (e) shows the time for computing the optimal routing probabilities with respect to the number of classes. The three products case for all 27 seeds requires approx. 5.5 minutes. If this is too much for interactive planning or a run of 40 different scenarios for a complete study of a volume program, just seeding the most important junctions, i. e. those at the bottlenecks, can help to reduce the optimization time while achieving the most effective results in this limited setting.

As stated above, this approach to determine optimal probabilities is guided by the idea to balance the work load and thus to minimize the maximum utilization within a class. This approach does not try to minimize lead time or work-in-process directly. But as it is shown in figures (f) and (g) the more classes are included in the optimization process the lower the work-in-process and the lead time, respectively, get. Thus in the example this approach reduces these measures as well.

An important point to mention is that the work-in-process and the lead time do not decrease monotonously. This is due to the fact that equalizing the utilization of a set of work centers at a junction on the one hand reduces the utilization of some work centers, but on the other hand for some others it might increase. Now it depends on the exact parameters (cycle times, batch sizes, etc.) of all work centers involved, whether the overall work-in-process is reduced or increased. Thus, the optimizer might fail to find an assignment that leads to minimum work-in-process or lead time, but it lowers the maximum utilization of the work centers involved. This might turn an overload situation into a configuration that allows the performance measures to be computed.

The last figure (h) shows the effect of this optimization procedure on the squared coefficient of variation of the utilization. The initial value is given for a the number of classes $x = 0$. This coefficient decreases due to the routing optimization as well.

The General Case

The approach above relies on the fact that the product mix does not change when the routings are changed so that the quadratic programming approach allows to exploit the structure of the automatically generated models. The problem in the general case is that the product mix α_l depends on the

number of visits at the sink operation of product l which in turn might depend on the routing probabilities at junctions. Thus if the yield is different in different branches of a junction, the objective (10.25) does not lead to an equal distribution of utilization because the product mix used to calculate the optimal solution might be different from the resulting product mix.

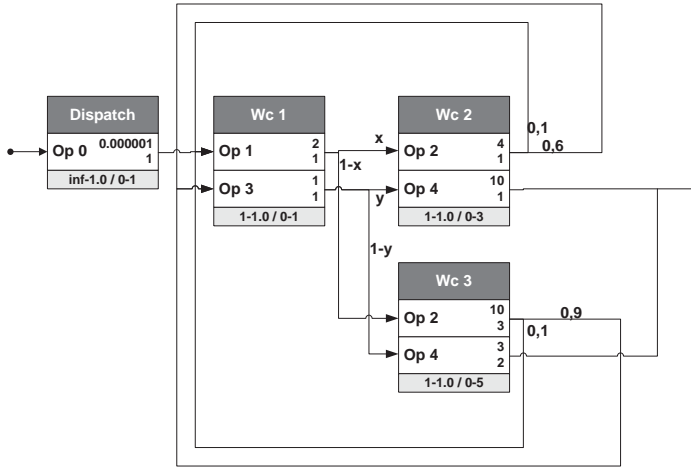


Figure 10.16.: Sample model

Some other general characteristics of junctions are self loops at junctions (i. e. the routing probability $p_{ii} \neq 0$ for a process step a_i that is either the root or an end-point of a junction), rework loops involving more process steps ending somewhere at a junction, and cascaded junctions (i. e. a branch of a junction branches again). All these characteristics can be handled by the objective in equation (10.25) as long as the product mix does not change. Figure 10.16⁴ shows a model that cannot be handled correctly by the approach described above as operation Op2 at Wc2 produces 30% scrap whereas Op2 at Wc3 does not produce any scrap at all. But the two degrees of freedom x and y allow to visualize the influence of routing probabilities on the performance measures. The construction of the bounds and constraints according

⁴See appendix D for an explanation of the parameters in the figure.

to equations. (10.23), (10.22), and (10.21) yield:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & : & 0 & 0 & 0 & 0 \\ -1 & 1 & -0.1 & -0.1 & 0 & 0 & 0 & : & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & : & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & : & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -0.6 & -0.9 & 1 & 0 & : & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & : & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & : & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & : & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & : & 0 & 0 & 1 & 1 \\ 0 & -0.001 & 0 & 0 & 0 & 0 & 0 & : & 1 & 0 & 0 & 0 \\ 0 & -0.001 & 0 & 0 & 0 & 0 & 0 & : & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.001 & 0 & 0 & : & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -0.001 & 0 & 0 & : & 0 & 0 & 0 & 1 \end{pmatrix} \bar{e} \leq \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \infty \\ \infty \\ \infty \end{pmatrix}$$

Figure 10.17 shows the results of a complete enumeration of the routing probabilities. Of course, being real numbers the probabilities cannot be completely enumerated. Instead equidistant samples are chosen from the interval $]0 \dots 1[$. The routes are assigned the following probabilities: There is a single product P . Define the following process steps: $a_1 = (\text{Op1}, \text{Wc1}, P)$, $a_2 = (\text{Op2}, \text{Wc2}, P)$, $a_3 = (\text{Op2}, \text{Wc3}, P)$, $a_4 = (\text{Op3}, \text{Wc1}, P)$, $a_5 = (\text{Op4}, \text{Wc2}, P)$, and $a_6 = (\text{Op4}, \text{Wc3}, P)$. The probability of the route (a_1, a_2) corresponds to the free parameter $x = p_{1,2}$ and that of the route (a_4, a_5) to $y = p_{4,5}$. Then the remaining routing probabilities are given by $\text{Prob}(a_1, a_3) = 1 - x = p_{1,3}$ and $\text{Prob}(a_1, a_5) = 1 - y = p_{1,5}$. Varying x and y within the range $]0 \dots 1[$ yields the following results.

Figure 10.17(a) shows the total utilization of the two work centers Wc2 and Wc3 . The utilizations are intersecting planes, which shows that there are junction configurations that allow for an equal distribution of utilizations. The mutual trade off of the work centers' load can be seen along both axes, i. e. by increasing x or y the utilization of one work center is increased and that of the other reduced.

Each junction configuration leads to different performance measures: The efficiency is shown in figure (b). Values of 0 as for $y = 0$ and $x > 0.8$ are due to the system being overloaded. An overview on the lead time of good parts is shown in figure (c). As $x \rightarrow 1$ the lead time of good parts increases until the system gets overloaded. In the latter case no points are shown in the diagram.

The surface that looks rather flat in figure (c) is zoomed into in figure (d) showing that there exists a minimum. How to deal with this general kind of problems is shown in the next section.

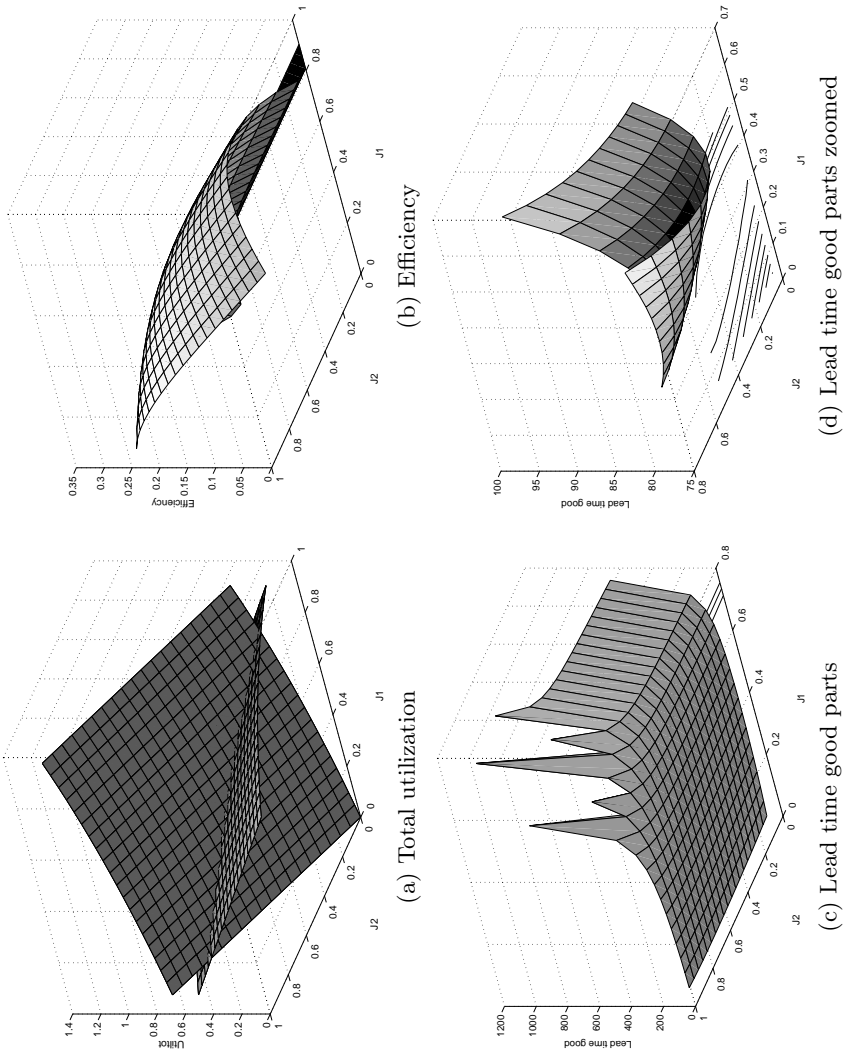


Figure 10.17.: Performance measures vs. routing probabilities of sample model

10.4. Optimization of Performance Measures by Evolutionary Search

The goal of this chapter is to show how analytic performance evaluation can guide genetic search towards promising scenarios to solve some selected production planning problems.

10.4.1. Simulation and Optimization

If exact mathematical methods fail to find optimal solutions in case of NP-hard problems often techniques from artificial intelligence (AI) are applied. Many researchers have followed the idea to combine heuristic search methods with simulation systems. Thus the ability of simulation to analyze complex dynamic systems and the power of the efficient search methods get combined. In these applications simulation serves as a means to evaluate the objective function. Examples of these approaches can be found in [Haj00, GP00, KW00, KB00, AFF00]. In [ZRV00] a *Great Deluge* algorithm [Due93] is applied for head count optimization. [Pfl96] distinguishes between the approach to take only objective function values into consideration (*black box*) and the approach that tries to estimate first or second derivatives for optimization (*white box*).

Unfortunately, optimization based on simulation is not as easy as it seems at first glance if random effects are included in the simulation. The following short discussion is taken from [RS93]. It helps to explain the difference between optimization with the help of discrete event simulation and queueing network analysis.

The optimization problem to solve for the decision vector v is a stochastic one

$$\min_{v \in V} l_0(v) = E_v[L_0(Y)] \quad (10.29a)$$

$$\text{s. to } l_j(v) = E_v[L_j(Y)] \leq 0, \quad j = 1, \dots, k, \quad (10.29b)$$

$$l_j(v) = E_v[L_j(Y)] = 0, \quad j = k + 1, \dots, M \quad (10.29c)$$

where $l_0(v)$ is the expected performance of a stochastic system and $L_0(Y)$ is the sample performance driven by an input vector Y having a cumulative distribution function $F(y, v)$. v is a vector of parameters lying in a parameter set $V \subset \mathbb{R}^n$ and Y represents the random effects in the model. The goal is to minimize the expected system performance l_0 under the stochastic equality

and inequality constraints equations (10.29b) and (10.29c), respectively. The subscript v in $E_v[L_0(Y)]$ means that the expectation is taken with respect to $F(y, v)$.

To estimate the performance l_0 through simulation for a fixed v a random vector Y_1 from the cumulative distribution function $F(y, v)$ is generated and $L(Y_1)$ is computed. Then a second random vector L_2 independent of L_1 is generated and $L(Y_2)$ is computed. After a repetition of N times $l(v)$ can be estimated by the sample mean $\hat{l}_N(v) = (1/N) \sum_{i=1}^N L(Y_i)$. By the strong law of large numbers $\lim_{N \rightarrow \infty} \hat{l}_N(v) = E[L(Y)]$ with probability one. Using standard statistical tools a central limit theorem and confidence intervals for $l(v)$ can be derived. Sensitivity analysis and optimization with respect to v based on this conventional statistical experimental design is rather time-consuming. In the late seventies two more sophisticated methods were introduced, perturbation analysis and the score function. Both of them permit estimation of all sensitivities from a single simulation run (experiment) in the course of evaluation of the performance measures. For further references see [RS93].

The important difference between discrete event simulation and queueing network analysis is that for the latter the objective $l_0(v)$ and the constraints $l_j(v)$ are computed analytically. No experiments and no statistical analysis have to be carried out. The problem (10.29) becomes a non-linear programming problem. Of course, general non-linear programming still is NP-hard, but the stochastic nature of the problem is already considered in the objective function. This section describes how the efficient methods of analytical performance evaluation and heuristic search methods can be combined. The problems considered in this section are based on simulation models that are used to derive parameter settings (the decision vector v) that lead to an improved overall system performance. The first task is to identify the planning questions that are to be answered. This includes the identification of decision variables which in turn define the parameter types for the optimization process.

The first two questions following have partly been answered, i. e. under special assumptions, by the methods explained in the previous sections.

- *Product mix.* In section 10.2 a product mix optimization is presented. The approach focuses on capacity and profit margins only. It does not take the performance measures like work-in-process or lead time into consideration.

The interesting question is how the solution of the linear program compares to a solution that takes the expected work-in-process or lead time into consideration.

- *Routing/load decision.* Routing optimization has already been discussed in section 10.3. But the approach presented there — equalizing the utilization of work centers at junctions — does not directly lead to optimal performance measures. Thus the interesting question is how the routings can be chosen to minimize the lead time, for example. Moreover, this allows to tackle the problems of the general case presented in the example in section 10.3.4.

Two other questions are also to be dealt with in this section:

- *Tooling.* Which tool set is needed to achieve a specified expected lead time or work-in-process level (goal programming)?
- *Investments.* If tool costs are given, which is the cheapest tool set to meet the predefined goal?

Again, these two topics include routing decisions: For example, if an investment should be made and a work center at a junction is to be added, the routings have to be adjusted. Thus reasonable routing probabilities are the key to models producing useful results.

These questions lead directly to the decision variables. Table 10.2 shows the parameters involved in these questions and their respective types. The

Variable	Type
Number of tools	integer
Demand rate	double
Routing probabilities	double

Table 10.2.: Decision variables

goal is to set up an optimization process over all degrees of freedom the model offers. An optimization problem that includes all degrees of freedom is a mixed-integer problem with constraints. The objective function is given by the performance measures computed by the queueing network.

Recall the sample model in figure 10.6 that cannot be optimized by the quadratic program developed in section 10.3. This is mainly due to the

secondary demand that depends on the routing probabilities. Moreover, the approach merely considers capacity but ignores the approximations of lead time and work-in-process computed by queueing theory formulae. To include these performance measures in the optimization an evolutionary algorithm is developed in this section.

The evolutionary algorithms are implemented as a client to the simulation server. Thus it can work on models created manually or on the generated ones. The algorithms are implemented in C/C++ and the simulation server is accessed via CORBA.

10.4.2. Chromosome Coding

Evolutionary algorithms are based on a representation of the solution to a problem. A chromosome for the problems considered here consists of a concatenation of all the parameters of the model that are decision variables. Table 10.2 shows that the parameters in question are numeric, i.e. double or integer values. The decision variables are either defined a priori (for example, the number of tools at a special work center) or are collected from the queueing model like the junctions that can be found automatically by the algorithm in figure 10.11. The corresponding genes for all degrees of freedom are automatically added to the chromosome.

In the evolutionary computing society there has been a long discussion on whether to prefer binary or real valued coding. [Gol89] started out with the so-called multi-parameter, mapped, fixed-point coding where all parameters are represented by a binary number that is linearly mapped onto the domain of the parameters. [Mic96] provides some experiments and reasons for the superiority of the real valued coding. The huge number of parameters of some optimization problems leads to unreasonably large chromosomes in the binary case. Moreover, the range of values is larger for the real case and it does not rely on a predefined resolution. Early experiments with the problems considered in the following showed as well that genetic search on binary coded chromosomes is inferior to real valued ones.

In general, a parameter is mapped directly to a gene. The parameters are represented by a class having the following attributes:

- *Name*. This can be used to identify parameters in the GA with their counterparts in the queueing model.
- *Type*. The type defines whether a parameter is of type double or inte-

ger. Some operators, like arithmetic crossover (see below), for example may produce a fractional value even if both genes involved are integer.

- *Domain bounds.* Each parameter i is to be chosen from a specific domain. Let u_i and v_i denote the lower and the upper bound, respectively. For example, routing probabilities fall in the range $]0 \dots 1[$, numbers of tools are non-zero and demand rates can be bounded from above by the maximum arrival rate.

All of the parameters shown in table 10.2 follow the one to one mapping approach, i.e. for every parameters there is exactly one gene. Moreover, for all parameters except for junctions the gene values correspond directly to the parameter values of the queueing model.

The routing probabilities at junctions are constrained. They have to be probability distributions, i.e. the probabilities of all possible routes at a junction and the scrap rate have to sum up to 1. From the constraint handling techniques shown in section 10.4.5 a special decoding approach is chosen as it allows the application of the normal genetic operators and avoids the generation and evaluation of infeasible solutions.

Let $j = (a_i, \mathcal{J})$ be a junction where $\mathcal{J} = \{a_{i_1}, \dots, a_{i_n}\}$. The junction j has $n - 1$ degrees of freedom which results in $n - 1$ parameters, one for each route except one. Given the scrap ratio s at the junction, these $n - 1$ probabilities uniquely define the probability distribution at the junction as

$$p_{ii_n} = P(a_i, a_{i_n}) = 1 - s - \sum_{k=1}^{i_n-1} P(a_i, a_{i_k}). \quad (10.30)$$

Thus each route except one in the junction corresponds to a gene c_i in the chromosome. The genes c_i do not hold the routing probabilities directly but rather a coded version: Let ψ_j be the junction probability of a junction j . Then the first route will get the probability $p_1 = \psi_j \cdot c_1$, the second one $p_2 = (1 - p_1) \cdot c_2$, etc. In general, the p_i are defined recursively as follows

$$p_1 = \psi_j \cdot c_1 \quad (10.31a)$$

$$p_i = \left(\psi_j - \sum_{j=1}^{i-1} p_j \right) \cdot c_i \quad \text{for } i = 1, \dots, n-1 \quad (10.31b)$$

$$p_n = \psi_j - \sum_{j=1}^{n-1} p_j. \quad (10.31c)$$

This allows to code one junction. The parameters for different junctions are simply concatenated in the chromosome like the other parameters. For example, the chromosome for a problem having l work centers, and two junctions with m and n degrees of freedom, respectively looks like

$$c = [\underbrace{c_1 \dots c_l}_{\text{num. of tools}} : \underbrace{c_{l+1} \dots c_{l+m}}_{\text{junction 1}} : \underbrace{c_{l+m+1} \dots c_{l+m+n}}_{\text{junction 2}}] \quad (10.32)$$

where c_1, \dots, c_l represent the number of tools and the remaining genes represent the routing probabilities. Both parts for junctions 1 and 2 are decoded according to equations (10.31). Which parameters are present in the chromosome depends on the problem to solve and its decision variables.

10.4.3. Initialization Schemes

The natural domains for the parameter values for routing probabilities are $[0 \dots 1]$. Note that the borders of the range are not feasible for the queueing network analysis thus $[0 + \epsilon \dots 1 - \epsilon] = [u \dots v]$ is used where ϵ denotes the smallest routing probability allowed and u and v are defined as the lower and upper bound, respectively. Junction optimization can make use of three different initialization schemes.

Firstly, all genes c_i of a chromosome can be initialized randomly, i. e.

$$c_i = u + r \cdot v \quad (10.33)$$

where r denotes a random number in the range $[0 \dots 1]$. This initialization is used directly as the coded gene value.

Secondly, the probabilities can be uniformly distributed

$$c'_i = \frac{\psi_j}{n}. \quad (10.34)$$

The chromosome of routing probabilities c' is coded according to the inverse procedure of equations. (10.31). This yields the chromosome c . It is a deterministic initialization. In order to avoid too many identical solutions the chromosome is randomized after the coding, i. e.

$$c_i = \begin{cases} u_i + r \cdot v_i & \text{with probability } p = 1/3 \\ c' & \text{with probability } p = 2/3 \end{cases} \quad (10.35)$$

replaces some genes with an arbitrary value from the parameter's domain. This is possible due to the decoding scheme that always generates a feasible routing probability distribution.

Thirdly, a local capacity related heuristic can help to find promising solutions

$$c_i = \psi_j \frac{C_k}{\sum_{j \in W_j} C_j} \quad (10.36)$$

where W_j is the set of all work centers that lie the at end-points of a junction, i. e. $W_j = \{w|j = (j, \mathcal{J}), a_l \in \mathcal{J}, w = w(a_l)\}$ and

$$C_k = \frac{r_k \cdot t_k}{\sum_{a \in \mathcal{A}_{u, w_k}} \frac{\mathbb{E}[S_a]}{b_K}} \quad (10.37)$$

is a local capacity measure for work center k . Note that in this case the number of tools at work center k is called t_k instead of c_k . Again, chromosomes initialized in this way are randomized by equation (10.35).

For the optimization of the probability distribution at junctions each chromosome is initialized with one of these initialization schemes with equal probability $p = 1/3$.

The second set of decision variables that gets a special initialization procedure is the demand rate. As in the extended linear program the demand specified in the queueing model is taken to be the lower bound u of the decision variables. The upper bound v_i is constructed according to the following procedure: Set the demands of all but one product to zero. Determine the maximum arrival rate $\hat{\lambda}_i^{\max}$ for the one remaining product i and use it as its upper bound v_i . Repeat this procedure for all products. This approach tries to bound the feasible region closely without losing any part of the search space. For the initialization a random demand is computed according to

$$c_i = u_i + r \cdot v_i \quad (10.38)$$

with a random number r from the range $[0, 1]$. For later use define

$$\hat{\lambda}^{\max} = \max_i \left\{ \hat{\lambda}_i^{\max} \right\} \quad (10.39)$$

for all products i .

The other decision variables, e.g. the number of tools and batch sizes, are bounded from below by $u = 1$ and from above by a constant C that can be arbitrarily large.

10.4.4. Genetic Operators

The evolutionary algorithm to develop is supposed to be suitable for integer and real parameters, so that it possible to use it for different planning tasks. In order to mix several decision variables easily like routing probabilities, demand rates, etc. common operators are examined that do not need to know what kind of parameter is positioned at which gene in the chromosome. Even the coded routing probabilities do not require any special treatment.

Crossover

Firstly, the crossover operators are examined. The operators for numerical optimization problems discussed in the following fall into three categories:

1. The traditional operators like one-point, two-point, n-point, and uniform crossover that swap the information on certain genes of both parents.
2. Arithmetic crossover operators that in a certain sense average the information stored on corresponding genes.
3. Heuristic crossover operators that perform little hill-climbing operations.

The traditional one-point and two-point crossover operators can be generalized to an n-point crossover. If n equals the length of the chromosome minus 1, the operator turns into the uniform crossover which decides for each gene according to an independent Bernoulli experiment from which parent the allele is to be taken. Calling the algorithm in figure 10.18 with the parameter $numXoverSites = 1, 2, n < length(c) - 1$, or the length of the chromosome minus 1, $length(c) - 1$, results in a one, two, n, or uniform crossover operator, respectively. These operators maintain feasibility with respect to the parameter domains.

In [Mic96] the need for a crossover operator allowing local fine tuning is pointed out. Therefore, a special operator — *arithmetical crossover* — is implemented. This operator constructs gene i of the child chromosome, c_i , by a linear combination of the two parent genes c_i^1 and c_i^2 , i. e.

$$c'_i = a \cdot c_i^1 + (1 - a) \cdot c_i^2 \quad \text{where } a \in]0 \dots 1]. \quad (10.40)$$

GENERAL CROSSOVER

Input: int *numXoverSites*, chromosome *c*₁, chromosome *c*₂**Output:** chromosome *c*₁, chromosome *c*₂**begin** *clen* = length(*c*₁); *l* = randomList(numXoverSites,1,*clen* - 1); // number, lower, upper limit append(*l*, *clen*); // avoids accessing uninitialized data in loop 2 sort(*l*); // increasingly **for** (*j* = 0; *j* < length(*l*); *j* = *j* + 2) { **for** (*i* = *l*[*j*]; *i* < *l*[*j* + 1]; *i* = *i* + 1) { swap(*c*₁[*i*], *c*₂[*i*])

}

}

return *c*₁, *c*₂**end**

Figure 10.18.: General crossover algorithm

This operator can either be applied on one single gene position or on the whole chromosome. In the latter case, the operator is the vector form of equation (10.40), i. e.

$$c' = a \cdot c^1 + (1 - a) \cdot c^2 \quad \text{where } a \in [0 \dots 1]. \quad (10.41)$$

As an example let $c_1 = [0.5 \ 0.5]$, $c_2 = [0.8 \ 0.8]$, $a = 0.1$, and $i = 2$. Then the offspring is $c'_1 = [0.5 \ 0.77]$ and $c'_2 = [0.8 \ 0.53]$. For $a = 0.9$ the offspring becomes $c'_1 = [0.5 \ 0.53]$ and $c'_2 = [0.8 \ 0.77]$. Note that if $a < 0.5$ the operator behaves more like a crossover, i. e. the order of the magnitude of the genes gets swapped ($(c_1)_2 = 0.5 < (c_2)_2 = 0.8$ and $(c'_1)_2 = 0.77 > (c'_2)_2 = 0.53$), and if $a > 0.5$ the order of the magnitude of the genes remains ($(c_1)_2 = 0.5 < (c_2)_2 = 0.8$ and $(c'_1)_2 = 0.53 < (c'_2)_2 = 0.77$). In the latter case the operator behaves more like a mutation operator.

Another distinction can be made on the choice of the parameter a . If a is constant, the operator is called *uniform* arithmetical crossover. It is called *non-uniform* if a is chosen at random. For example, a can be reduced with the age of the population.

Thus, to summarize there are four different arithmetical crossover operators: single gene uniform, chromosome uniform, single non-uniform, and

chromosome non-uniform.

If the parameter domains are intervals, then these four operators maintain feasibility with respect to the domain constraints – not necessarily with respect to other constraints like feasibility (no overload) of the simulation model.

The last two operators incorporate a kind of local search into the crossover operation. The first one is called *heuristic* crossover [Mic96]. Without loss of generality, let $z(c_1) < z(c_2)$ (or $z(c_1) > z(c_2)$ in case of a maximization problem), i.e. c_1 is the better chromosome. Let the offspring c' be

$$c' = r \cdot (c_1 - c_2) + c_1 \quad (10.42)$$

where r is a random number in the range $]0 \dots 1[$.

Note that this operator uses in contrast to the others the objective function value to determine the direction of the search. Moreover, it produces either 1 offspring or none: If an infeasible offspring is constructed, the operator is re-applied until a feasible solution is found. If after a maximum number of applications no feasible solution is found, the operator fails and does not produce any offspring at all.

In order to use it with the existing frame work, the operator is adjusted in two ways: Firstly, choosing two different random parameters $r_1 \in [0 \dots 1]$, $r_2 \in [1 \dots 2]$ leads to two children. The r_i , $i = 1, 2$, correspond to two different step widths in a hill-climbing step. Secondly, if a parameter leaves its domain, it is projected onto the nearest border by setting $c_i = \max\{c_i, u_i\}$ or $c_i = \min\{c_i, v_i\}$. Thus, feasibility with respect to domain restrictions is maintained and the operator returns two feasible chromosomes. This avoids a perhaps unsuccessful search for a feasible solution.

Experiments showed that often few genes caused a rather good fitness of a chromosome but a lot of other genes defined the wrong search direction despite the fact that $z(c_1) < z(c_2)$. Thus, a new operator — *heuristic random crossover* — performs a Bernoulli experiment for each gene to determine the search direction. Again, the probability of choosing the wrong direction, i.e. the direction $(c_2 - c_1)$ at gene i , can decrease with the age of the population. Adapting the parameter a with respect to the age of the population can emphasize exploration in earlier generations of the evolutionary algorithm whereas in later stages local fine tuning is realized.

Mutation

Mutation operators are supposed to perform small changes in a chromosome to bring back alleles into a population that have been lost. The basic mutation operator *uniform* mutation selects a parameter value randomly from the parameter's domain.

This might be too disruptive, thus producing too many bad solutions if the population has evolved over a long period. To get around this problem, [Mic96] suggests an operator called *non-uniform* mutation. It is especially useful for local fine-tuning.

The non-uniform mutation operator realizes an adaptive control of the randomness. Let $c = [c_1 \dots c_i \dots c_n]$ be a chromosome and c_i was selected for mutation. Mutation leads to the chromosome $c' = [c_1 \dots c'_i \dots c_n]$ where

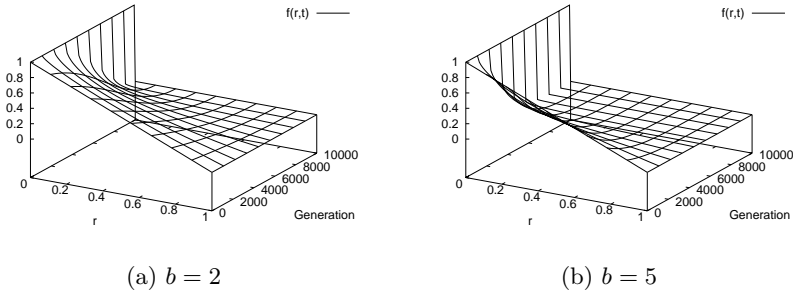
$$c'_i = \begin{cases} c_i + \Delta(t, v_i - c_i) & \text{if } X = 0 \\ c_i + \Delta(t, c_i - u_i) & \text{if } X = 1 \end{cases} \quad (10.43)$$

u_i, v_i denote the lower and upper bound of the domain of the parameter, respectively, and X is a Bernoulli random variable with $P(X = 0) = 0.5$ and $P(X = 1) = 0.5$. The function $\Delta(t, y)$ returns a value in the range $[0, y]$ so that the probability of $\Delta(t, y)$ being close to 0 increases as t increases. This property causes the operator to search the search space uniformly in the beginning of evolutionary search (when t is small), and very locally at later stages; thus increasing the probability of generating the new number closer to its successor than a random choice. [Mic96] suggests

$$\Delta(t, y) = y \cdot \left(1 - r^{\left(1 - \frac{t}{T}\right)^b}\right) \quad (10.44)$$

where r is a random number from the interval $[0 \dots 1]$, T is the maximum generation number, and b is a system parameter determining the degree of dependency on the iteration number ($b = 5$ is suggested). Figure 10.19 shows the function $f(r, t) = 1 - r^{\left(1 - \frac{t}{T}\right)^b}$ for different values of b .

Both operators, the uniform and the non-uniform mutation, could be either applied to a single gene or to the whole chromosome. However, uniform mutation on the whole chromosome would mean a complete random re-initialization of the chromosome. Thus this case is not considered. This leaves three mutation operators to examine uniform mutation, single-gene non-uniform mutation, and chromosome non-uniform mutation.


 Figure 10.19.: $f(r, t)$ for $b = 2$ and $b = 5$

10.4.5. Constraint Handling Techniques

Recall the optimization problem in equation (10.1). Let $x \in S$ be an element of the search space. Let

$$S' = \{x : x \in S, C(x) \in Q\} \quad (10.45)$$

be the set of all feasible solutions satisfying the constraints. In general, the structure of S' is not known, for example S' might be non-contiguous or non-convex.

The following methods can be applied to solve constrained optimization problems with evolutionary algorithms:

1. Problem specific genetic operators

If a feasible initial population can easily be generated, then such a population can be chosen and carefully designed operators assure that only feasible offspring is generated. This is only possible if the structure of S' is known. Unfortunately this is not often the case.

2. Penalty Functions

Many problems have a lot of constraints and finding a feasible solution is almost as difficult as finding the extremum. Thus infeasible solutions should be considered as well to guide genetic search. Often, optimum solutions lie on the border of S' , so that there are solutions that are infeasible but nevertheless more similar to the optimum than

many feasible solutions. In this case the problem can be solved by the relaxed problem 10.46. Constraint violation is considered in the objective function by so-called penalty terms:

$$\min_{x \in S} z(x) + \sum_{j=1}^n \Phi_j(\delta_j(C(x)_j, Q_j)). \quad (10.46)$$

Here Φ_j are monotonously increasing penalty functions where $\Phi_j(0) = 0$ for all $j = 1, \dots, n$ where n denotes the number of constraints. $\delta_j(C(x), Q_j)$ is a metric function measuring the distance of $C(x)_j$ to Q_j , $\delta_j(C(x), Q_j) = 0$, if $C(x) \in Q_j$. Φ_j adapts the objective function value $z(x)$ in dependence of the amount of constraint violation measured by δ_j . For example, in the case of $C(x) \in \mathbb{R}$ and $Q \in \mathbb{R}$, δ_j can be chosen to be the squared distance and Φ is a multiplication with a constant, i.e. a linear penalty function. Choosing a special Φ_j and δ_j allows a different weighting of each constraint violation.

Under these assumptions an optimum solution x^* to the problem (10.1) is also a solution to the relaxed problem, as $\Phi_j(\delta_j(C(x), Q_j)) \geq 0$. Moreover, each optimum solution to the relaxed problem is a lower bound for the optimal value of $z(x^*)$ of the original problem, which is normally better than an optimal value of $z(x)$ ignoring the constraints.

Dynamic penalty functions are presented in [ST93]. Φ is proportional to the difference between the best feasible and the best overall objective function value. A thorough analysis of feasibility, fitness landscapes, and penalty functions for the knapsack problem can be found in [Got01].

Penalty functions are especially useful for large feasible search spaces. For heavily constrained problems finding suitable Φ_j and δ_j can be a very difficult task and depends on the problem. The use of penalty functions is not restricted to evolutionary algorithms. They play also an important role in non-linear programming [PSU88].

3. Decoding Schemes

This approach decodes the information stored in the chromosomes so that only feasible solutions are constructed. For example, consider the decoding scheme for routing probabilities presented in section 10.4.2

that constructs a feasible probability distribution for n routings from $n - 1$ real values.

For this approach it must be possible to decode every possible individual and each decoded solution has to be feasible. The decoding algorithm has to be efficient (in the complexity theoretical sense) and continuous, i. e. small changes in the coded solution should only cause small changes in the decoded solution. The problem of this approach is the dependency on the optimization problem and the difficulty to find a suitable coding that satisfies the requirements mentioned above so that the genetic evolution converges to an optimum.

4. Repair Algorithms

This approach is suitable for problems that allow for an easy repair of infeasible solutions, i. e. that an infeasible solution can easily be adapted to become feasible while staying similar to the infeasible one. If this is possible either the objective function value of the repaired individual is used for fitness evaluation or the repaired individual replaces the infeasible one. The repair method can either be a greedy algorithm, a random change, or a special heuristic.

This approach is problem-specific and for some problems repairing an infeasible solution is as difficult as solving the original problem, for example of time-tabling or non-linear transport problems. Moreover, the problem arises that the genotype of the repaired individual does not necessarily resemble the original genotype, so that the building blocks cannot be inherited as assumed.

5. Two-phase GA

In [SX93] a problem-independent approach to solve general constrained optimization problems is presented.

In a first phase a random initial population for the second phase is generated. A random population is evolved with a usual EA where the objective is to minimize the amount of violation of the first constraint. Then the resulting population is used to minimize the violation of the second constraint, etc. Individuals violating previous constraints are assigned a fitness value of 0. In this phase actions have to be taken that the population does not converge but spreads equally across the search space.

In the second phase the original objective function is optimized starting with the final population of the first phase as the initial population. Again, infeasible individuals are assigned a fitness value of 0 and are thus eliminated by the selection procedure.

The advantage of this method is its independence from the problem to solve and from the fitness of infeasible individuals. Moreover, the second phase optimizes the original objective function and no transformation (with penalties etc.) thereof.

A detailed discussion on constraint optimization and finding feasible solutions can be found in [Mic96, p. 312ff], [MF00, p. 231ff], and [Got99]. The latter distinguishes between *direct* and *indirect* search in the feasible search space. The direct methods include repair algorithms together with local search heuristics whereas the decoding schemes belong to the class of indirect approaches.

In the tasks of optimizing the performance values of the queueing networks the following constraints arise:

- *Junction constraints.* The sum of routing probabilities leaving a junction is constant $1 - s$ where s denotes the scrap ratio.
- *Parameter domains.* Each parameter has a natural domain, e.g. a number of tools is greater than 0, a routing probability smaller than 1, etc.
- *Overload border.* Capacity planning shall enforce that the maximum utilization stays below 100%. Performance measures can only be evaluated for this case by queueing theory.

In section 10.4.2 a decoding scheme has been presented for the routing probabilities that assures that every chromosome corresponds to a feasible solution with respect to the probability distribution at junctions. The genetic operators presented assure that no parameter values are generated that fall outside the parameters domain. Handling the overload border is the most difficult task. This is tackled by penalty terms in the objective functions presented in the next section.

10.4.6. Objective Functions

The last topic to deal with are the objective functions. Three different planning topics will be examined in this section, each having a special objective

function:

1. Routing optimization
2. Product mix
3. Investment analysis

As mentioned above, the main idea is to find optimal parameter configurations with respect to the performance measures calculated by the simulation server.

Routing Optimization

The first planning task requires the lead time of good parts to be minimized. As the analytic performance evaluation is not able to calculate this measure in the overload case, the goal function is not defined completely on the cartesian product of all parameter ranges, the search space S . Thus the objective function is extended by a penalty term based on the maximum arrival rate that allows to measure how far an infeasible solution is away from the set of feasible solutions. This allows to guide the optimization process towards feasibility.

This leads to the following goal function:

$$\min z(x) = \begin{cases} l_g & U_{\max} \leq 1 \\ C + 10000 \cdot (a - a_{\max}) & U_{\max} \geq 1 \end{cases} \quad (10.47)$$

where l_g denotes the lead time of good parts as calculated by the queueing network analyzer, C a constant that is larger than the maximum value of the performance criteria to be minimized (lead time of good parts in this case), and a and a_{\max} are the arrival rate and the maximum arrival rate, respectively. Whereas a is constant for a constant yield, a_{\max} may vary with modified routings. As the routing probability distribution approaches the feasible set, the goal function decreases towards its lower bound C .

This goal function is used for the experiments 1 and 2 that try to determine the best genetic operators for the optimization of the routing probabilities.

Product Mix

In order to test the genetic algorithm for the suitability of the product mix optimization the performance of the algorithm is tested on the normal optimal product mix problem described in section 10.2. Of course, the linear program presented there is doing the job much faster and it is no use trying to solve problems for which efficient algorithms are available by heuristic search methods. However, as a benchmark this problem is tried. The advantage is that an optimum solution to compare with can easily be computed.

The goal function makes use of two different performance measures, the profit p and the maximum total utilization u_{\max} of the simulation model. Formulated as a minimization problem the goal function becomes

$$z(d) = \begin{cases} C - p^r & \text{for } u_{\max} \leq U_{\max}^* \\ C \cdot ((1 - \varrho_{\max}^*) + u_{\max}) & \text{else} \end{cases} \quad (10.48)$$

where d is the decision vector, d_i is the demand of product i , and u_{\max} is the maximum utilization of the model, C denotes a constant for which $C > p^r$ must hold. $\varrho_{\max}^* < 1$ is a constant defining the maximum utilization allowed in the production line. This constant is defined a priori. C can be set to $C = (\hat{\lambda}^{\max} c^{\max})^r$ where $\hat{\lambda}^{\max}$ is defined in (10.39) and c^{\max} is the maximum contribution margin of all products. C helps to distinguish feasible and infeasible solutions. The exponent r defines the slope of the objective function. The larger r is the larger are the differences between individuals of different fitness. However, setting r too high might lead to premature convergence. Experiments showed that $r = 3$ is sufficient to yield satisfying results.

The first branch of the objective function is the actual optimization problem, i. e. maximizing the profit, whereas the second branch is the penalty function that helps to guide the algorithm towards the feasible region.

Figure 10.20 shows the structure of the goal function for a sample model. The model examined has got four different products p_1, \dots, p_4 . Thus the goal function in this case is $z(d_1, d_2, d_3, d_4)$. In the optimal solution determined by the linear program $d_1 = d_3 = 0.01$, i. e. these two products stay at the lower bound. The graph shows the function $z(0.01, d_2, 0.01, d_4)$.

The left wing corresponds to the first case of equation (10.48), the right wing to the second. This is a typical example of an optimization problem for which the optimum solution can be found on the border of the feasible region.

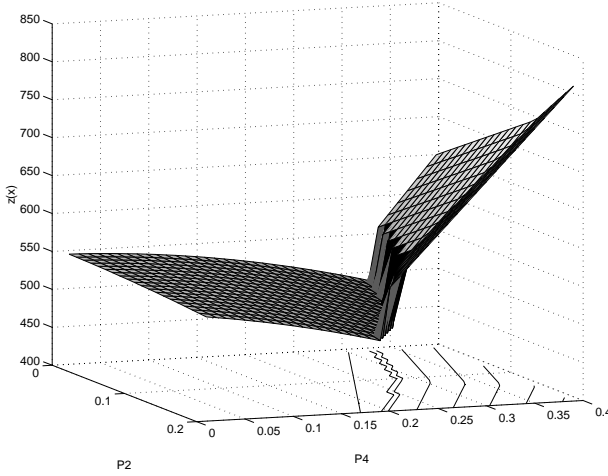


Figure 10.20.: Goal function including penalty term for $r = 2$

Note that the border is characterized by the long valley which decreases at a very small slope.

To go beyond the analysis of the linear program, the objective function of equation (10.47) is used for the evolutionary algorithm to find a product mix that is optimal with respect to the lead time of good parts l_g . In order to avoid trivial solutions a further constraint is needed. This is a minimum throughput T . The problem becomes

$$\min z(d) = l_g \quad (10.49a)$$

$$\text{s. to } \sum_{i=1}^U d_i \geq T \quad (10.49b)$$

$$\varrho^{tot} < 1.0 \quad (10.49c)$$

where d is the decision vector, $d_i, i = 1, \dots, U$, is the demand of product i , and ϱ^{tot} is the maximum total utilization of the model which is equal to the total utilization of the first bottleneck.

The objective function (10.49a) is not used directly, as this is a constrained optimization problem: Constraint (10.49b) guarantees a minimum

throughput whereas (10.49c) assures that the queueing model is computationally feasible. The constraints are modeled as penalty terms. As shown in section 10.4.5 the design of a proper penalty function is a difficult task. In this case two measures of different magnitudes have to be compared: the total utilization and the sum of the demand rates. Both can be used to measure the amount of infeasibility. To avoid the problem of handling different magnitudes of constraint violation, the constraint violation for constraint i , v_i , is computed as

$$v_i = \frac{\text{actual value of measure } i}{\text{goal value of measure } i} \quad (10.50)$$

if the actual value has to be less than the goal. Otherwise, the violation is measured by the reciprocal value. Thus the following two constraint violations can be computed

$$v_1 = \begin{cases} \frac{\varrho^{tot}}{0.99} & \text{if } \varrho^{tot} > 0.99 \\ 0 & \text{else} \end{cases} \quad (10.51)$$

and

$$v_2 = \begin{cases} \frac{T}{\sum_{i=1}^U d_i} & \text{if } \sum_{i=1}^U d_i < T \\ 0 & \text{else} \end{cases}. \quad (10.52)$$

v_1 and v_2 can then be used to formulate the objective function for the evolutionary algorithm

$$z(d) = \begin{cases} l_g + \alpha v_1 + \beta v_2 & \text{if } \varrho^{tot} < 0.99 \\ C + \alpha v_1 + \beta v_2 & \text{else} \end{cases} \quad (10.53)$$

where C is a constant that should be larger than the lead times of good parts. Note $l_g \rightarrow \infty$ as $\varrho^{tot} \rightarrow 1$. But l_g stays finite due to constraint (10.51). α and β are two constants of approximately the same size. Together with v_1 and v_2 they define the penalties and guide the genetic search towards the feasible region if necessary.

Investment Analysis

The third objective is to determine the most effective spots for further investments. Given a production system and a predefined budget, the question

is which tools are to be bought to improve the overall system performance. The performance measure in this case is the lead time of good parts. Of course, this is bounded from below by the raw process cycle time of good parts, i.e. adding infinitely many tools does not drive the lead time towards 0. Reducing the overall lead time means improving the overall efficiency as the raw process cycle time of good parts stays almost fixed — *almost*, because the raw process cycle time may vary with the routings.

This task leads to the following goal programming problem:

$$\min_{p_{ij}, c_k} \sum_{k=1}^{|W|} c_k C_k \quad (10.54a)$$

$$\text{s. to} \quad \sum_{\substack{j=(a_i, \mathcal{J}) \\ a_j \in \mathcal{J}}} p_{ij} = \psi_j \quad j \in J \quad (10.54b)$$

$$p_{ij} \geq \epsilon \quad j = (a_i, \mathcal{J}), a_j \in \mathcal{J}, j \in J \quad (10.54c)$$

$$l_g \leq L \quad (10.54d)$$

$$\rho^{tot} < 1.0 \quad (10.54e)$$

where the p_{ij} are the routing probabilities at junctions, c_k is the of number of tools at work center k , C_k is the cost of installing one tool at work center k , l_g is the lead time of good parts as determined by the queueing model, ρ^{tot} is the maximum utilization as determined by the model, and ϵ is the minimum routing probability allowed. The special case of setting $C_k = 1$ for all work centers leads to a problem neglecting the tool cost and taking only the number of tools into consideration.

Constraint (10.54b) assures that the routing probabilities at the junctions of a queueing model are probability distributions, (10.54c) defines minimum routing probabilities, (10.54d) specifies the lead time goal, and (10.54e) assures that the system is not overloaded.

The demand is defined a priori. The product mix might change as changes in the routing might modify the product yield and thus the secondary demand. Decision variables are the number of investments c_k per work center and the routing probabilities p_{ij} , because adding a tool at a junction requires a new calculation of the distribution.

For this task the chromosome contains two sections: one for the number of tools and one for the routing probabilities:

$$c = [c_1 \dots c_n :: r_1 \dots r_l] \quad (10.55)$$

where the $c_k, k = 1, \dots, n$, denote the number of tools for the work centers and the r_i denote the routing probabilities. For the number of tools there is a one to one mapping between gene values and the corresponding parameter of the simulation model. The routing probabilities are decoded by the method described in equations. (10.31).

Initialization of the number of tool part is done randomly within the domain. A heuristic could define an upper bound like $c_k \cdot 10$, for example, where c_k denotes the number of tools necessary to make an arbitrary initial configuration feasible. The routings are initialized as described in section 10.4.3, i.e. either randomly, uniformly distributed, or heuristically based on the work centers' capacities at the end points of junctions.

Again, the objective function (10.54a) is modified to handle the constraints: The lead time of good parts and the utilization are handled by penalty terms. As for the previous goal, the optimum throughput constrained product mix, the constraint violations v_1, v_2 for the utilization and the lead time goal, respectively, are computed. The probability distribution at junctions (constraints (10.54b) and (10.54c)) is handled by the decoding scheme described in section 10.31. Thus the modified problem becomes

$$\min_{p_{ij}, c_k} \sum_{l=1}^{|\mathcal{W}|} c_k \mathcal{C}_k + \alpha v_1 + \beta v_2 \quad (10.56)$$

where

$$v_1 = \begin{cases} \frac{q^{tot}}{0.99} & \text{if } q^{tot} > 0.99 \\ 0 & \text{else} \end{cases} \quad (10.57)$$

and

$$v_2 = \begin{cases} \frac{l_g}{L} & \text{if } l_g > L \\ 0 & \text{else} \end{cases}. \quad (10.58)$$

This goal function is examined in the fourth experiment.

10.4.7. Experiments

This section presents the results of the application of the operators and objective functions defined in the previous section.

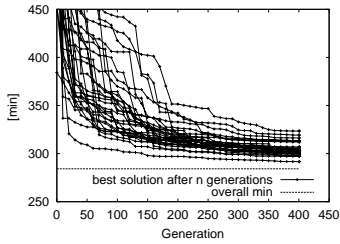
Experiment 1: Crossover Operator for Routing Optimization

This experiment focuses on the crossover operators for routing optimization. The ten operators presented in section 10.4.4 are examined in detail. The test problem is a problem consisting of four different products, ten work centers, 68 operations and 100 routings. The degree of freedom for all junctions is 32, thus the chromosomes have got a length of 32. For all experiments the modified GA by Michalewicz [Mic96] is applied whose implementation has already proved good performance in [Pau96] and [Mee97]. The EA parameters are set as follows: population size = 30, linear scaling with a scaling factor of 2. For each generation 10 individuals are selected for the application of the genetic operators. Each run is stopped at 400 iterations and for each of the 10 crossover operators 30 independent runs are started. This results in an overall number of $10 \cdot 10 \cdot 30 = 3000$ trials to find the minimum lead time of good parts and $3000 \cdot 400 = 1200000$ fitness evaluations. The mutation rates for the three mutation operators described in section 10.4.4 are kept rather low so that the GA performance is not mainly due to the mutation. The rate for the three mutation operators is set to 0.05 each.

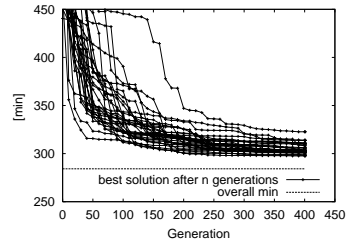
Figures 10.21 and 10.22 show the results of each of the 10 operators' runs. These graphs give a first impression on the performance of the different operators. The x -axis represents the number of generations, the y -axis shows the lead time of good parts. Common to all graphs is that the performance measure (lead time good parts) decreases monotonously. This is due to the elitist strategy that assures that the best chromosome is kept within the population when the next generation is constructed.

The main points to focus on are at the end at iteration 400 in each graph: The lower and the closer the goal function values are together, the better. For example, comparing figures (i) and (j) shows that the random heuristic operator (j) produces lower values with less variance. Moreover, it gets to better solutions more quickly than the heuristic crossover operator (i). This can be seen from the fact that the overall curve looks more like a straight angle. Whereas the traces in figures 10.21 and 10.22 only allow for a quantitative impression, the statistics shown in table 10.3 offer a qualitative analysis. The table shows the number of runs per operator ($N=30$) in each case, the mean, the standard deviation, the standard error, the 95% confidence interval, and the minimum and the maximum value.

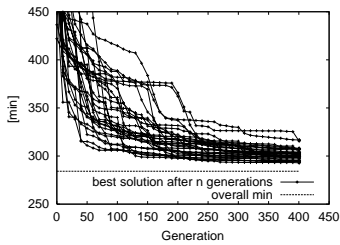
The best solution is found by the random heuristic crossover operator (284.2). Moreover, this operator yields the smallest standard deviation, er-



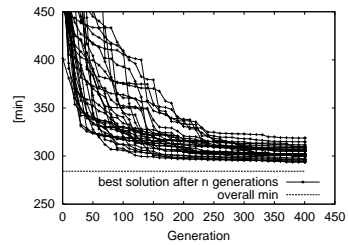
(a) One-point crossover



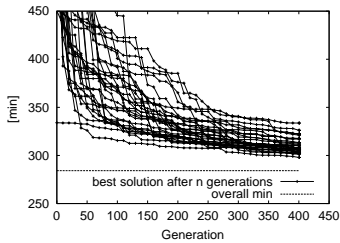
(b) Two-point crossover



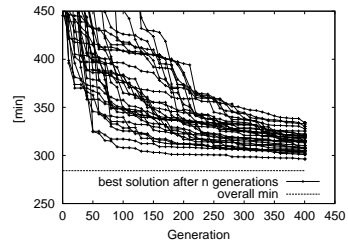
(c) n point crossover



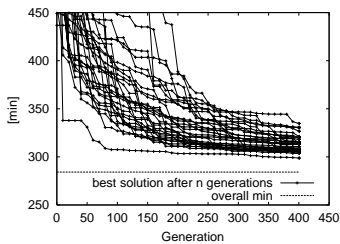
(d) uniform crossover



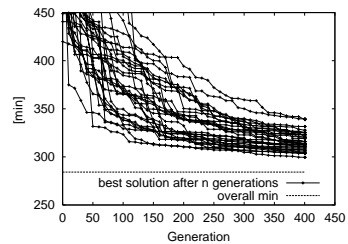
(e) Arithmetic single gene uniform



(f) Arithmetic single gene non-uniform



(g) Arithmetic multiple gene uniform



(h) Arithmetic multiple gene non-uniform

Figure 10.21.: Experiment 1: Traces (Operators 1–8)

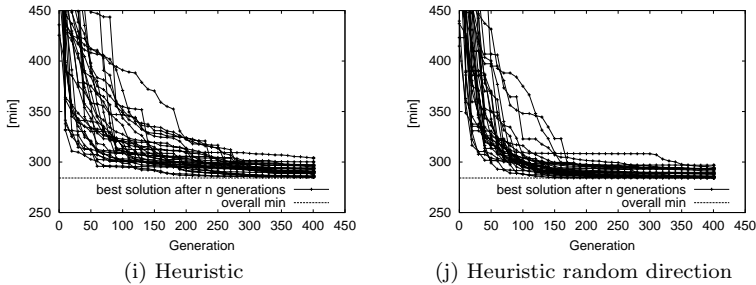


Figure 10.22.: Experiment 1: Traces (Operators 9 and 10)

	N	Mean	St.dev.	St.err.	95% conf.int.		Min	Max
					LB	UB		
One-point	30	304.8977	7.0127	1.2803	302.2791	307.5162	291.65	323.46
Two-point	30	304.9160	5.4295	0.9913	302.8886	306.9434	297.59	322.67
n-point	30	301.8030	5.8244	1.0634	299.6281	303.9779	292.96	316.74
Uniform	30	303.1483	6.0789	1.1099	300.8784	305.4182	293.56	318.70
ArSingleU	30	310.2617	8.3062	1.5165	307.1601	313.3632	297.90	333.68
ArSingleNU	30	314.6600	9.7953	1.7884	311.0024	318.3176	296.32	334.13
ArMultipleU	30	313.6773	8.8455	1.6150	310.3744	316.9803	298.78	334.77
ArMultipleNU	30	316.2110	10.0448	1.8339	312.4602	319.9618	299.55	339.57
Heuristic	30	291.4473	4.7303	0.8636	289.6810	293.2137	284.93	303.99
RandHeu	30	288.4223	4.0598	0.7412	286.9064	289.9383	284.20	296.86
Total	300	304.9445	11.4596	0.6616	303.6424	306.2465	284.20	339.57

Table 10.3.: Experiment 1: Descriptive statistics

ror, and the smallest confidence interval. This is summarized in figure 10.23 that shows the minimum, average, and maximum lead time of good parts for each operator numbered from 1 to 10.

Whether there is a statistically significant difference between the operators is examined by an analysis of variance (ANOVA). An analysis of variance is an overall test to check if the means of trials are significantly different [Har93]. The ANOVA results for experiment 1 are shown in table 10.4. The results show that there is a significant difference between the means of the

	Sum squares	Deg.Frd.	Mean squares	F	Sig.
Between the groups	23823.400	9	2647.044	49.712	0.000
Within the groups	15441.675	290	53.247		
Total	39265.075	299			

Table 10.4.: Experiment 1: ANOVA

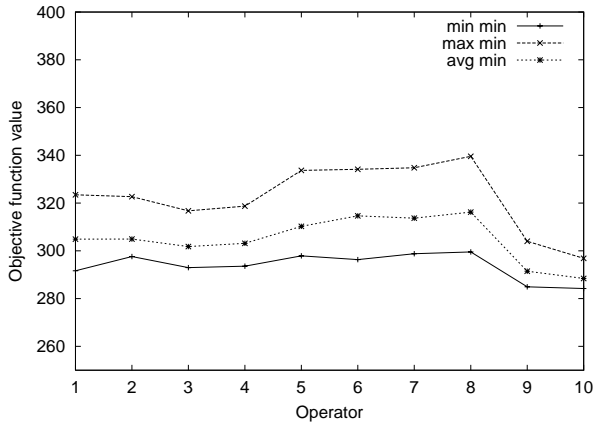


Figure 10.23.: Experiment 1: Overall results

performance of the 10 crossover operators. To determine which trials are in fact different, multiple comparisons are needed. This can be done by the tests of Scheffé or Tukey [Har93, p. 616]. Table 10.5 shows the results of the Scheffé test, i. e. the means of the groups forming homogeneous sub-groups. It shows that both heuristic crossover operators (Heuristic and RandHeu) perform significantly best. Both operators cannot be distinguished under the assumptions of the test, although figures 10.22 (i) and (j) and the statistics showing lower mean, minimum, maximum and deviation might suggest that the latter performs better. In addition, the RandHeu operator is the

		N	Subgroups for $\alpha = 0.01$			
	Crossover		1	2	3	4
Scheffé procedure	RandHeu	30	288.4223			
	Heuristic	30	291.4473			
	n-point	30		301.8030		
	Uniform	30		303.1483		
	One-point	30		304.8977	304.8977	
	Two-point	30		304.9160	304.9160	
	ArSingleU	30		310.2617	310.2617	310.2617
	ArMultipleU	30			313.6773	313.6773
	ArSingleNU	30				314.6600
	ArMultipleNU	30				316.2110
	Significance		0.978	0.020	0.012	0.357

Table 10.5.: Experiment 1: Post hoc analysis (Scheffé)

	N	Mean	St.dev.	St.err.	95% conf.int.		Min	Max
					LB	UB		
M1 (UM)	60	292.5698	16.8398	2.1740	288.2196	296.9200	284.34	383.03
M2 (SGNUM)	60	287.9542	3.5794	0.4621	287.0295	288.8788	284.18	305.08
M3 (CNUM)	60	290.6008	3.9572	0.5109	289.5786	291.6231	284.45	302.71
Total	180	290.3749	10.3175	0.7690	288.8574	291.8925	284.18	383.03

Table 10.6.: Experiment 2: Descriptive statistics

operator that finds the best overall solution that is not achieved by any other operator.

Roughly speaking, the next group of operators are the traditional cross-over operators (one, two, n point, etc) which outperform most of the arithmetical operators (apart from the single uniform) operator. According to [Mic96] the arithmetic crossovers are a good means for local fine-tuning. As an overall approach they seem to be outperformed at least in this test setting.

Experiment 2: Mutation Operators

In order to examine the performance of the three mutation operators — uniform (UM), single gene non-uniform (SGNUM), and chromosome non-uniform (CNUM) — a similar test is performed. In this case the modified genetic algorithm by Michalewicz is used again with 10 selected individuals for each generation. The population size is 30, linear scaling is applied with a factor of 2.0, and each run is stopped at a maximum of 400 generations. For these experiments the most promising parameter settings from experiment 1 are used, i.e. the random heuristic and the two-point crossover operators are applied at rates of 0.6 and 0.1, respectively. These two operators are selected so that the best one and one operator of the second best class are taken.

Figure 10.24 shows the traces of the optimization runs. The traces suggest the best performance of the non-uniform mutation operator applied to the complete chromosome. Especially the uniform mutation operator got stuck in local optima more frequently. It looks as if the first two operators need more time to converge to the optimum.

For the qualitative analysis table 10.6 shows the descriptive statistics of experiment 2. The best overall performance value is found by the second mutation operator 284.18, even slightly better than the best solution of ex-

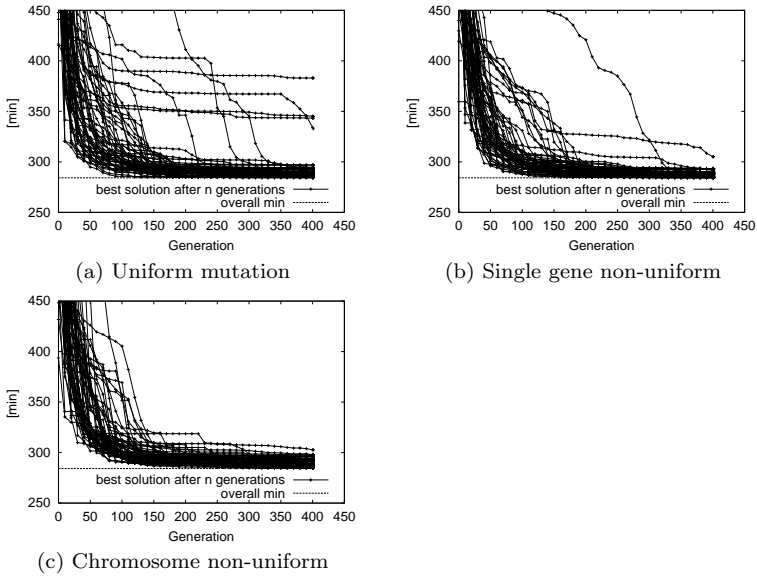


Figure 10.24.: Experiment 2: Traces

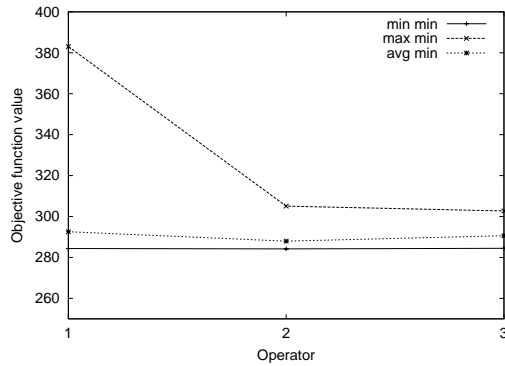


Figure 10.25.: Experiment 2: Overall results

	Sum squares	Deg.Frd.	Mean squares	F	Sig.
Between the groups	643.724	2	321.862	3.094	0.048
Within the groups	18410.995	177	104.017		
Total	9054.719	179			

Table 10.7.: Experiment 2: ANOVA

		N	Subgroups for $\alpha = 0.05$	
	Crossover		1	2
Scheffé procedure	M2 (SGNUM)	60	287,9542	
	M3 (CNUM)	60	290,6008	290,6008
	M1 (UM)	60		292,5698
	Significance		0.366	0.573

Table 10.8.: Experiment 2: Post hoc analysis (Scheffé)

periment 1 (284.20). However, the two other mutation operators are not far off. The second operator also achieves the best average of all runs, the smallest error and deviation, and the smallest confidence interval. Moreover, the interval is not included in that of the third operator. Figure 10.25 visualizes the overall results (minimum, average, maximum) of the three different mutation operators.

Again, the ANOVA allows to test whether there are significant differences in the means. Table 10.7 shows the analysis of variation of experiment 2. At a level of $\alpha = 0.05$ there is a significant difference between the means. Application of Scheffé's procedure yields the results shown in table 10.8. Both non-uniform operators perform better than the non-uniform one. However, none of them is found to be superior.

Experiment 3: Product Mix Optimization

The first part of this experiment examines the performance of the operators on the product mix problem explained in section 10.2. Thus the optimal solution can be computed with the help of the linear program and can be compared with the solutions the evolutionary algorithm finds.

The second part applies the operators to the problem of finding an optimal product mix with respect to lead time and a throughput goal (equation (10.49)). For both parts the modified GA by Michalewicz is applied with a population size of 20 and 8 individuals selected for the application of

the operators at each generation. Linear scaling with a factor of 2.0 is used as well as the elitist strategy. The mutation rates are 0.5 for each of the mutation operators. The crossover rate is 0.8 for each of the ten operators in turn.

Ten operators are considered. These are one-point, two-point, uniform, arithmetic single uniform, arithmetic single non-uniform, arithmetic multiple uniform, arithmetic multiple non-uniform, heuristic, random heuristic, and a combination of random heuristic and arithmetic single uniform, single non-uniform, chromosome uniform in the ratio 5:1:1:1. The operators are numbered from 1 to 10. Note that as the test model contains only four decision variables, n -point crossover for $n = 3$ results in a uniform crossover and is thus not applied in this setting.

The traces of the EA runs can be found in figures 10.26 and 10.27. Common to all traces is that the EA converges quite fast within the first hundred generations.

	N	Mean	St.dev.	St.err.	95% conf.int.		Min	Max
					LB	UB		
One-point	30	493.8573	1.9747	0.3605	493.1200	494.5947	488.72	498.23
Two-point	30	493.2313	1.7602	0.3214	492.5741	493.8886	488.06	496.11
Uniform	30	493.7423	2.5119	0.4586	492.8044	494.6803	488.24	498.98
ArSingleU	30	494.3927	2.5157	0.4593	493.4533	495.3320	487.72	499.56
ArSingleNU	30	494.4327	2.3824	0.4350	493.5431	495.3223	488.17	498.13
ArMultipleU	30	495.7410	2.7429	0.5008	494.7168	496.7652	489.50	501.25
ArMultipleNU	30	494.9853	2.6440	0.4827	493.9980	495.9726	491.20	501.39
Heuristic	30	486.2247	1.3494	0.2464	485.7208	486.7286	485.65	490.52
RandHeu	30	486.2547	1.1369	0.2076	485.8301	486.6792	485.65	489.64
RandHeuArith	30	487.2170	1.8293	0.3340	486.5339	487.9001	485.65	491.90
Total	300	492.0079	4.2086	0.2430	491.5297	492.4861	485.65	501.39

Table 10.9.: Experiment 3: Descriptive statistics

The descriptive statistics are shown in table 10.9. The optimum solution (485.65) is only found by the last three operators, followed by the traditional crossover operators, and the arithmetic ones. The combination of the random heuristic and the arithmetical crossover could not improve the performance of the EA. The overall performance is summarized in figure 10.28. Each integer value from 1 to 10 on the x-axis belongs to one of the ten operators. The performance measures shown are the minimum, average, and maximum solution of each of the 30 runs per operator. In addition, the optimum solution is shown that is only reached by operators 8, 9, and 10.

Both operators 8 (Heuristic) and 9 (RandHeu) find the optimum solution. However, on the one hand, operator 8 yields an average minimum that is slightly smaller than that of operator 9. On the other hand, operator 9

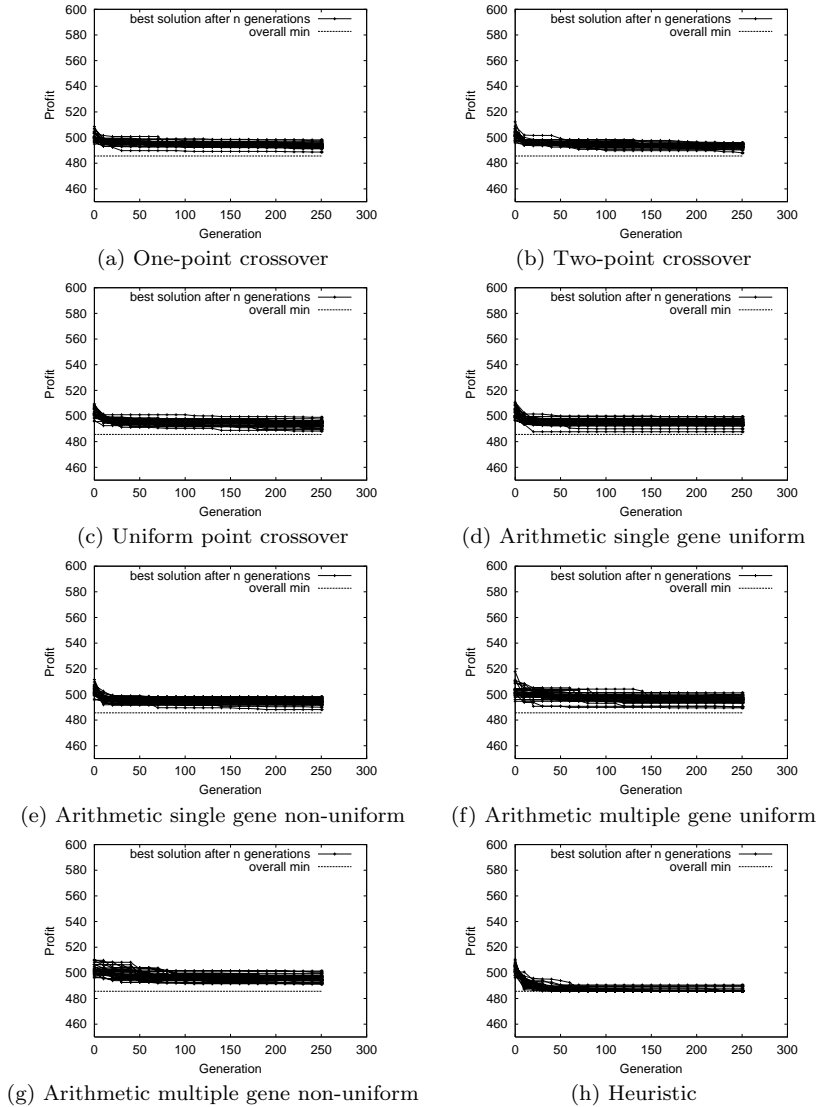


Figure 10.26.: Experiment 3: Traces (Operators 1–8)

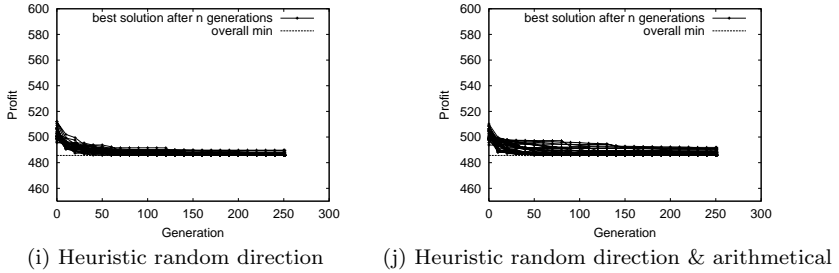


Figure 10.27.: Experiment 3: Traces (Operators 9 and 10)

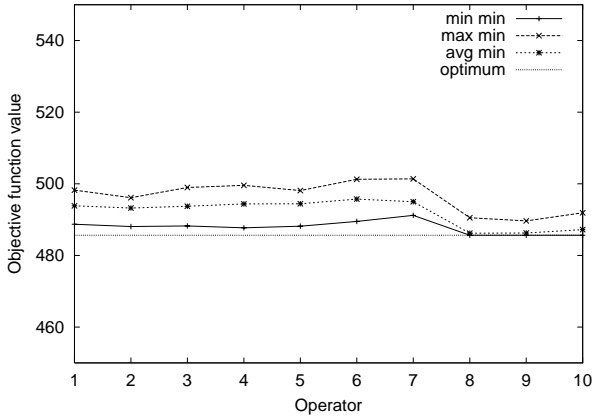


Figure 10.28.: Experiment 3: Overall results (profit)

yields the better maximum minimum. With respect to these measures, both operators perform best.

The optimum solution computed by the linear program is $d = (0.010000, 0.105479, 0.010000, 0.277264)$. This is exactly the same solution found by operators 8, 9 and 10.

The ANOVA results are shown in table 10.10. There is a significant difference between the operators. Scheffé's procedure finds the subgroups. The results are shown in figure 10.11. The small differences between operators 8, 9, and 10 are not significant. However, these three perform better than the

	Sum squares	Deg.Frd.	Mean squares	F	Sig.
Between the groups	3953.742	9	439.305	94.911	.000
Within the groups	1342.296	290	4.629		
Total	5296.038	299			

Table 10.10.: Experiment 3: ANOVA

Crossover	N	Subgroups for $\alpha = 0.05$		
		1	2	3
Heuristic	30	486.2247		
RandHeu	30	486.2547		
RandHeuArith	30	487.2170		
Two-point	30		493.2313	
Uniform	30		493.7423	493.7423
One-point	30		493.8573	493.8573
ArSingleU	30		494.3927	494.3927
ArSingleNU	30		494.4327	494.4327
ArMultipleNU	30		494.9853	494.9853
ArMultipleU	30			495.7410
Significance		0.955	0.357	0.171

Table 10.11.: Experiment 3: Post hoc analysis (Scheffé)

rest of the operators examined in this experiment.

For the second part of the experiment, all of the above operators are now applied to the problem of determining the optimal product mix with respect to a minimum lead time of good parts and a throughput target (equation 10.49). This experiment is summarized in figure 10.29. As for the other experiments, the operators 8, 9, and 10 perform best. All three of them find the same optimum solution. Operator 9 yields the best average minimum and the best maximum minimum, followed by operator 8 and operator 10.

Experiment 4: Investment Analysis

This experiment examines the suitability of the operators presented for integer optimization. The decision variables are the number of tools at each work center. The EA parameters are set as follows: The size of the population is 30, 10 individuals are selected for replication in each generation of the modified GA by Michalewicz, 15 runs for each setting, mutation rates of 0.1 for each of the three mutation operators, crossover rate 0.7 for each of the ten operators in turn, and linear scaling and elitism. The trials are run for a maximum of 500 generations. After 100 generations without improvement the EA is considered to be converged and the run is interrupted to save time.

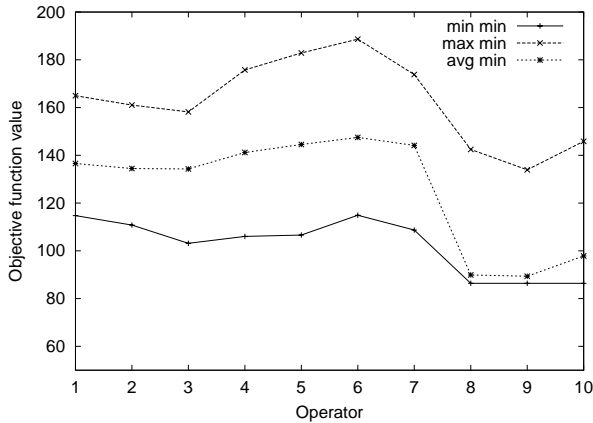


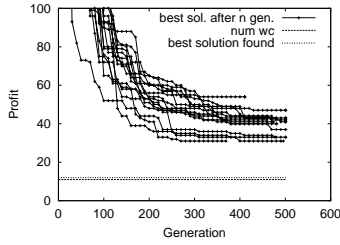
Figure 10.29.: Experiment 3: Overall results (throughput goal)

	N	Mean	St.dev.	St.err.	95% conf.int.		Min	Max
					LB	UB		
One-point	15	40.1333	6.3117	1.6297	36.6380	43.6287	31.00	54.00
Two-point	15	32.6000	5.9016	1.5238	29.3318	35.8682	22.00	46.00
n-point	15	29.2667	4.0438	1.0441	27.0273	31.5061	23.00	38.00
Uniform	15	26.8000	3.6095	0.9320	24.8011	28.7989	21.00	33.00
ArSingleU	15	49.4000	6.5661	1.6954	45.7638	53.0362	41.00	66.00
ArSingleNU	15	55.5333	9.6796	2.4993	50.1729	60.8937	41.00	75.00
ArMultipleU	15	71.5333	9.4708	2.4453	66.2886	76.7781	55.00	86.00
ArMultipleNU	15	65.8000	12.7794	3.2996	58.7230	72.8770	38.00	83.00
Heuristic	15	13.8667	0.9904	0.2557	13.3182	14.4151	12.00	16.00
RandHeu	15	13.4667	0.6399	0.1652	13.1123	13.8211	12.00	14.00
Total	150	39.8400	20.5204	1.6755	36.5292	43.1508	12.00	86.00

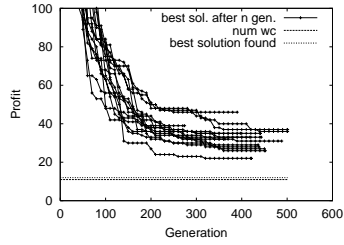
Table 10.12.: Experiment 4: Descriptive statistics

The traces are shown in figures 10.30 and 10.31 together with the best solution found and the initial configuration that violated the lead time constraint (num wc). Again, the best performance can be expected from the two heuristic crossover operators (i) and (j). In these traces a new phenomenon can be seen: The algorithms with traces shown in (a) to (h) get stuck in local optima quite frequently. This can be seen from the fact that several runs are interrupted at generations smaller than 500.

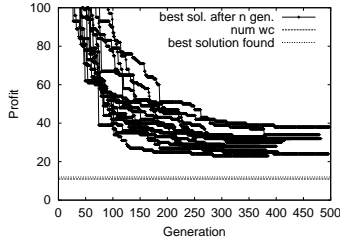
The descriptive statistics are shown in table 10.12. The minimum solutions are only found by the two heuristic crossover operators, of which the RandHeu operator achieves the smaller maximum solution. These operators



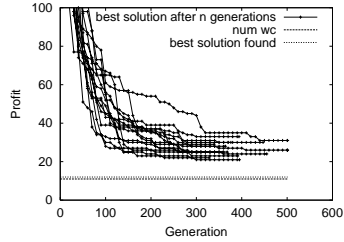
(a) One-point crossover



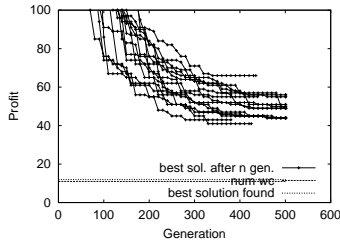
(b) Two-point crossover



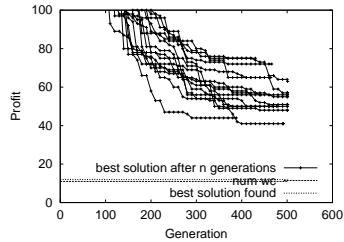
(c) n point crossover



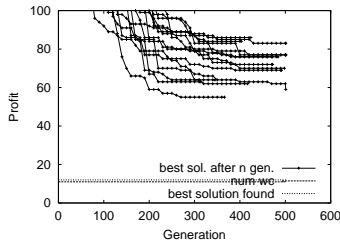
(d) uniform crossover



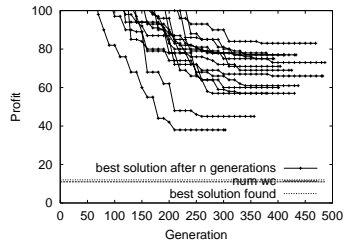
(e) Arithmetic single gene uniform



(f) Arithmetic single gene non-uniform



(g) Arithmetic multiple gene uniform



(h) Arithmetic multiple gene non-uniform

Figure 10.30.: Experiment 4: Traces (Operators 1–8)

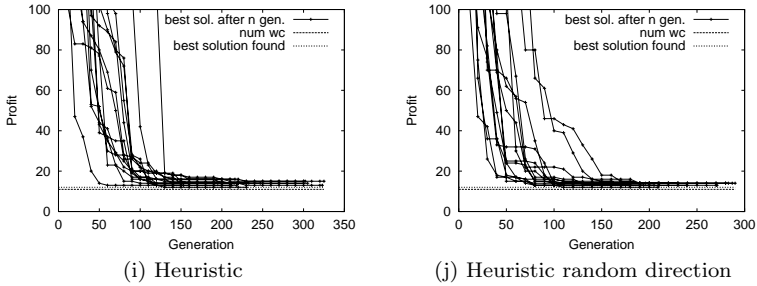


Figure 10.31.: Experiment 4: Traces (Operators 9 and 10)

	Sum squares	Deg.Frd.	Mean squares	F	Sig.
Between the groups	55808.560	9	6200.951	125.207	0.000
Within the groups	6933.600	140	49.526		
Total	62742.160	149			

Table 10.13.: Experiment 4: ANOVA

are followed by the traditional crossing operators. The worst minimum is found by the ArMultipleU operator and the largest standard deviation can be observed for the ArMultipleNU operator.

The overall results of the experiment are summarized in the graph in figure 10.32. For each of the ten crossover operators the performance measures (minimum, average, maximum goal function value) of the evolutionary algorithm are shown. Additionally, the best solution found (12) and the number of tools (11) that was assigned previously but violated the lead time constraint are shown.

The ANOVA results are shown in table 10.13. As can be seen from the last column, there is a significant difference within the operators. To find them, Scheffé's procedure is applied again. The results are shown in figure 10.14. Outstanding performance in this setting is achieved by the two heuristic crossover operators, RandHeu and Heuristic, followed by the traditional ones. But they are too far off to be acceptable, more than 200% of the best solution with respect to the average and minimum approximate solution.

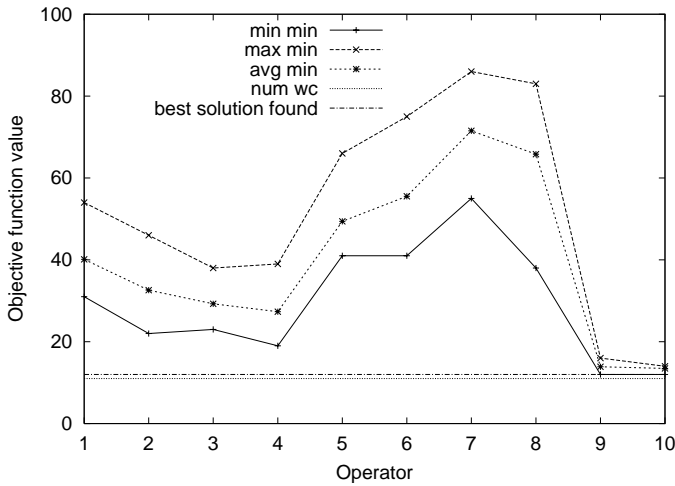


Figure 10.32.: Experiment 4: Overall results

Crossover	N	Subgroups for $\alpha = 0.05$						
		1	2	3	4	5	6	7
RandHeu	15	13.4667						
Heuristic	15	13.8667						
Uniform	15		26.8000					
n-point	15		29.2667					
Two-point	15		32.6000	32.6000				
One-point	15			40.1333				
ArSingleU	15				40.1333			
ArSingleNU	15				49.4000	49.4000		
ArMultipleNU	15					55.5333	55.5333	
ArMultipleU	15						65.8000	65.8000
Significance		1.000	0.823	0.480	0.175	0.767	0.078	0.833

Table 10.14.: Experiment 4: Post hoc analysis (Scheffé)

10.4.8. Results

The experiments showed that evolutionary algorithms are well suited for the optimization of decision variables of $G/G/c/\infty$ queueing networks. Having the EA framework at hand all that leaves to solve a new optimization problem is to specify a new goal function. Several samples have been presented. The most difficult task is the design of the objective function including penalty terms for the constraints. The queueing network analysis provides a good means to guide genetic search. Even in the case of overload situations, the maximum utilization or the maximum arrival rate show the direction towards feasible solutions.

Concerning the operators, three classes of operators are examined. The traditional crossover operators that swap the information of one or several genes. The second class are the arithmetical operators that are applied either to a single gene or to the whole chromosome. They compute weighted averages of the gene values. The last class are the heuristic crossover operators (Heuristic, *RandHeu*). The first of these is presented in [Mic96] and performs a small hill-climbing step in the direction of the better individual. The observation that often single genes with a strong influence on the overall performance of the individual lead towards the wrong direction motivated the operator *RandHeu*. This operator decides separately for every gene on the search direction. This yields better performance on the minimum, average, and maximum minimum found for most problems.

The second best class are the traditional operators. The arithmetical operators could not convince on the problems tested. The idea to combine them with the heuristic operators to let the heuristic operators do the exploration of the search space and let the arithmetic ones do the local fine tuning did not improve the situation as shown by the third experiment.

In general, evolutionary algorithms can help to find decisions on the basis of $G/G/c/\infty$ queueing network analysis as shown by the objective functions and operators presented in this section. The most promising operators for this kind of problem are the heuristic crossover operators.

10.5. Conclusion

This section summarizes the results and contributions of chapter 10 Optimization and presents some visions on further planning tasks and methods to be included in the EPOS system.

10.5.1. Results

This chapter introduced different optimization tasks and techniques that are required by systems for production planning. The main problems discussed are planning the manufacturing output, routing optimization/load balancing, and investment analysis.

Two different approaches to optimization tasks are shown: On the one hand, mathematical programming approaches that are based directly on the object model of the simulation, and on the other hand evolutionary algorithms that use the queueing network to evaluate the goal function of the optimization process.

The mathematical programming approaches are integrated directly into the queueing network analyzer. This allows the simultaneous calculation of the queueing network's performance measures and of the optimal product mix, for example. Based on the same model, no additional efforts are necessary to copy data from one model into another, and automatically created models can be used as well as interactively created ones.

The genetic algorithm approach shows how the queueing network analysis can be used to guide genetic search. Thus the strengths of two worlds are combined: Genetic algorithms are efficient search methods that have shown their usefulness in many applications. As search algorithms they rely on a huge number of objective function evaluations. This requires a fast evaluation of the objective, which is provided by the queueing network analysis: Discrete event systems require either long runs to allow the determination of renewal cycles or a large number of runs with an additional statistical analysis including multiple comparison procedures to determine parameter settings that differ significantly. In contrast to them queueing network analysis directly calculates the expected performance measures so that the optimization of queueing networks performance measures reduces to a deterministic nonlinear constrained optimization problem.

The contributions of this chapter are the following. By using the EPOS system as a basis for the queueing model and optimization, it is possible to use one common model non-redundantly for queueing network analysis and the optimization tasks. This is especially important as one of the most difficult tasks is to gather accurate data for computational models. Integrating the optimization algorithms with the collaboratively maintained models, it has been shown that the latter can serve as a basis for further optimization tasks.

The models and algorithms presented in this chapter are applied to large real-world models. For example, the model of the wafer manufacturing line contains nearly 20,000 objects, i. e. work centers, routings, operations, products.

The product mix optimization in section 10.2 is an extension of the classic model to meet the demands of the EPOS processes. Additional parameters are added like the lower bound of demands that allows to assure that certain predefined requirements are met or the maximum utilization level that can be read from a line profile and assures to control the resulting lead time.

Moreover, the product mix optimization is completely integrated into all planning processes: On the one hand, the scheduled simulation calculates the optimal mix for each scenario of the volume plan. The specified demand is taken as the lower bound. On the other hand, interactive planning with the EPOS Analyzer allows a quick analysis of different scenarios, ignoring the lower bounds for example by specifying different demands.

As the product mix optimization is based on the queueing network model it can benefit from the results. The computation of the number of visits for the process steps is essential to determine the correct load on the work centers. Thus scrap and rework rates of the generated model can be taken into account. Furthermore, all complex modeling steps applied during the automatic model generation are reflected in the product mix algorithm as well, because the algorithms work directly on the queueing model on the simulation server.

The quadratic program for routing optimization/load balancing allows a fast determination of routing probabilities at junctions of a queueing network. This helps to reduce the maximum utilization of certain work centers which might turn an infeasible queueing model, i. e. $\varrho^{tot} > 1$, into a feasible one. The formulation as a quadratic program and the fast solution procedure of the IBM optimization subroutine library allow to solve this problem during the automatic analysis of a complete volume program as well as for interactive planning. The decision variables arise naturally as on the way from the shop-floor-control database to the queueing network several modeling steps are necessary. The definition of the probability distributions, however, remains unspecified. The algorithm uses this natural freedom to make the utilization of work centers at the end-point of a junction as equal as possible.

Unfortunately, this approach cannot be applied to models having different scrap rates in different branches of a junction. Changing the routing

probabilities in this case leads to a different secondary demand that makes the problem more difficult. To solve this problem an approach based on an evolutionary algorithm has been established. It allows to define the routing probabilities so that a performance measure of the queueing network, for example the lead time of good parts, is minimized.

Moreover, other planning tasks are tackled by evolutionary algorithms, as well. This is the first time that evolutionary search has been applied to the optimization of $G/G/c/\infty$ queueing network decision variables. Several techniques are presented to handle the constraints: a decoding scheme to handle routing constraints and penalty functions to handle lead time and utilization goals. Thus direct search and indirect search are combined in the same individuals.

Ten different crossover operators have been presented, the last one being a modification of a heuristic crossover operator from literature. This newly defined operator performed at least as well as the original one, and it leads to better performance on some of the problems considered. The evolutionary algorithms are implemented as a client to the EPOS simulation server. Thus all kinds of models — automatically generated and interactively created ones — can be used for the optimization algorithms.

10.5.2. Outlook

The EPOS system offers a lot of possibilities for extensions with respect to optimization methods. Having most of the relevant data and the IT infrastructure to obtain additional parameters in a distributed manner at hand, it is possible to integrate other planning tasks from operations research and production management. Two very important tasks of production planning are lot-sizing problems and scheduling techniques which both seem suitable to answer the questions arising on the operational planning level.

Lot-sizing problems determine optimal production quantities for multi-stage production systems with respect to capacity, production, and inventory cost [Hel94, Tem95, Kle00b]. These problems are closely related to the sourcing problem, the question of when to start which products in which production line. As the EPOS simulation server can hold several models, it is a good start for a detailed analysis of such problems. Having the simulation models of the production lines of each stage at hand allows to implement optimization strategies that take several production lines into consideration. The model provides cycle times, batch sizes, process flows etc. The ques-

tion of how far lot-sizing and sourcing algorithms can profit from queueing network analysis has to be discussed.

Another modeling approach for the operational level tries to answer the question which parts in a production line are to be processed when at which machines in order to reduce the work-in-process as fast as possible or in order to meet due dates and other requirements. For these questions, fluid models seem to offer an answer. The approximation of discrete stochastic networks by fluid models to derive scheduling strategies involves three steps: The first one sets the theoretical framework, i.e. it discusses the type of approximation and convergence. A general introduction to fluid models is [CM94]. It formally describes the limit theorems of fluid models.

The second step is to derive the optimal fluid schedules. A first approach to the fluid scheduling problem of multi-class networks can be found in [CY93]. This paper describes an iterated linear programming approach that tries to drain an initial amount of work-in-process (fluid) at minimum cost. This approach has been extended in [CFY94] to set up schedules that are optimal over a time period defined a priori, like a shift. Both papers rely on trajectories that are optimal at every iteration. More elaborate techniques have been proposed by Gideon Weiss and others. These overall optimal draining strategies are explored in [DW99, Wei95]. This approach has been shown to work for rather small models. However, these are computationally more complex and expensive than the linear programming approaches.

The third step is to transform the optimum solution to the fluid scheduling problem back into a scheduling policy for the underlying discrete stochastic network. This is subject of [Har96, Mag00, Mag99].

All data needed for a fluid model analysis is available in the EPOS system. The construction of the fluid model from the EPOS model is quite straight-forward. The buffers in the fluid model literature correspond to the operations (process steps) in EPOS. This approach yields approximately 8000 buffers for the wafer model. Thus the question arises how sensible cost functions can be defined that are suitable to help practitioners in their daily planning tasks of a manufacturing line.

Another direction of further possible research is that of having the genetic algorithm not only to optimize certain parameter vectors, but to help in the design of manufacturing systems. This is a strategic planning problem. Taking system design, i.e. cellular, flow line, or job shop layouts, as decision variables might help to find the optimal design for a manufacturing system.

The design of manufacturing cells has already been discussed in [Mee97], for example. However, optimizing the design of a manufacturing system under consideration of the performance measures will be a demanding task.

Chapter 11

Integration with Shop-Floor-Control

11.1. Introduction

Integrated simulation as discussed in previous parts of this work integrates simulation into the *daily business* of production planning. Formerly the maintenance of simulation models, i.e. keeping the model on an acceptable level of validity has been a tedious, time-consuming and costly work. *Integrated simulation* provides the framework to facilitate, formalize, and speed-up the different tasks involved in maintaining a simulation model.

The permanent ability to use simulation as a planning tool offers new possibilities and changes the role that shop-floor-control (SFC) systems play in conjunction with simulation. Whereas traditionally these systems have been utilized to once parameterize simulation models with gathered data this task has now to be formalized into an ongoing process yielding new applications which have formerly not been accessible to simulation.

This chapter presents the types of applications that become possible when simulation is integrated with actual data from the shop floor. One application of each type is presented in detail showing possible chances and potential problems in the creation and usage of the new applications.

11.1.1. Overview

When combining data from simulation and shop-floor-control systems the following four generic types of applications are possible:

1. *Extraction of model input parameters.* Parts of the structure of the simulation model, i.e. the objects of the model and their interdependencies can often be loaded more or less directly from the databases of the shop-floor-control systems. The Parameters of the model have to be estimated from collected actual data.
2. *Problem detection.* By analyzing differences between plan figures from simulation and actual data measured by the shop-floor-control system, potential problems can be detected while filtering out normal, but problem-alike situations.
3. *Generation of operational plans.* By using current data from the shop floor the simulation model can be used to calculate values for operational production planning, i.e. results which can be used by line control.
4. *Validation of the simulation model.* Comparing actual data against the outcome of the simulation helps in detecting possible errors in the model.

An application of the first type has already been introduced in part 2 where the upload of structural data from the shop-floor-control system is presented. This includes process flows, operations, sectors, etc. This information forms the structure of the simulation model. The parameters of simulation objects like process times, availabilities, first time yields, etc. are unknown and can only be estimated. For each parameter Θ an estimator $T_{\Theta}(x_1, \dots, x_n)$ using historical data in form of the time series x_1, \dots, x_n is applied yielding an estimate $\hat{\Theta}$ for the unknown parameter Θ . It has to be kept in mind that the estimate $\hat{\Theta}$ is a random variable and thus the calculated performances measures of any analytical model become random variables (see [GH74, p. 344]).

According to Schwinn [Sch93, p. 90] problem detection is to a great extent supported by controlling plans which are currently realized. A new planning cycle is triggered by differences between planned and actually realized figures. This common truth is also valid for comparing simulation

results and data from the shop floor whereas in this case simulation provides the planned measures while the actual value is calculated using data from the shop-floor-control system. Due to the vast amount of stochastic influences in the production processes a comparison between plan and actual is a statistical test

$$H_0 : x_{plan} = x_{actual} \quad (11.1)$$

$$H_1 : x_{plan} \neq x_{actual} \quad (11.2)$$

with H_0 as the null-hypothesis specifying that the plan and actual values are equal and H_1 as the alternative. A test that results in choosing (under some significance level α) the alternative *detects* some kind of problem. Holding on to H_0 means there is not enough statistical evidence to detect a problem. This feature of a statistical test is very valuable for problem detection systems as it works like a filter. In general, the probability of having a difference between plan and estimated value equal to zero is itself zero. It is therefore the task of a statistical test to detect the differences which are large enough to present real problems. An application of this type using the notion of quality control charts is presented in section 11.2.

Loading current data from SFC systems allows the simulation model to become accessible for operational planning. So far, *integrated simulation* has been discussed as a decision support system for tactical production planning. This implies that parameters of the simulation model are planned values. This makes sense for tactical scenarios, e.g. even though a planned target value cannot yet be reached in the daily operation, it could be reasonable to specify this value because it serves as a target which could be reached within the scope of the planning horizon. For the operational use of the simulation model these planned values can create a problem as they might not reflect the current situation. Only if a simulation model reasonably reflects the current situation, it can be used to calculate targets on the operational level. Applications which configure a simulation model to reflect the current situation thus enable the use of the model in operational scenarios. An example of this type of application is the forecast of work-in-process presented in section 11.3.

Before simulation models can be applied to detect problems or to generate forecasts the model has to be validated. These tasks are greatly facilitated by the use of the quality control charts developed in section 11.2 which

led to the insight that batch sizes in the model had to be adjusted. To automate this process two features have been developed which are presented in section 11.4.

11.1.2. The Goal

Integrating shop-floor-control and simulation — as presented in this chapter — aims at creating a decision support system, a system that supports the production planner by delivering easy to maintain yet powerful reports that offer insight in the often complex logistical processes found in today's production lines.

It must be pointed out that this integration does not try to provide a fully automated line control which would replace the production planner or line controller. In section 11.3, for example, a forecast of the expected date of finish of the current work-in-process and an algorithm that matches the expected outcome with the demand is presented. It would be possible to automatically prioritize batches on basis of their individual positive or negative lateness calculated by the algorithm. Even though the result of the matching algorithm proves to be very valuable for the line control staff a fully automated control would leave out some important constraints which have not yet been included in the simulation model due to complex and time-consuming maintenance of the data needed, e.g. machine configurations, set-up states, priorities, customer-specific product designs, etc.

11.1.3. Review of the Literature

This section reviews literature published on the subject of integration of simulation and shop-floor-control systems. Most of the available sources deal with just one part — either the analysis of actual performance measures taken from shop-floor-control systems or simulation and the statistical inference of queueing systems. The literature to be presented in the following can be classified into the areas (i) logistical benchmarks, (ii) production management of semiconductor manufacturing, (iii) statistical process control (SPC), (iv) statistical inference of queueing systems, and (v) design and control of queues.

Logistical Benchmarks

References in this area deal with logistical performance measures which are normally seen as part of a wider system of key performance measures. This also implies that these sources treat logistical performance measures from a rather high level point of view (see [Web93], [Göp93a], [Göp93b], [Hei93], [CIM91, p. 140], [BGR94], [Jün89, p. 98]).

The current situation concerning the measurement of benchmarks is discussed controversially. Weber [Web95, p. 19] notes open problems in measuring process-related performance measures: A survey showed that only 34% of the companies questioned measured the lead time of produced orders which stands in sharp contrast to the fact that 84% of these companies consider this measure as important. The VDI guideline [CIM91, p. 59] complains that the available data hardly satisfies logistical purposes: As a rule, the data is quantitatively and qualitatively insufficient, not up-to-date, and often not suitable.

The situation in technology production lines appears to be different. Leachman and Hodges [LH97] compare performance measures of 28 wafer fabrication facilities located in the US, UK, Germany, Spain, Japan, Korea and Taiwan. Among numerous other benchmarks their findings include that all production sites apply SPC to their processes and equipment and that leading sites utilize computerized tracking systems to achieve

- excellent data collection and excellent data analysis capabilities,
- measure the overall equipment efficiency of their key processing equipment in order to identify losses in throughput,
- automatically capture the equipment status from machine logs to compare processing time against engineering standards.

According to the article leading sites utilize automated production planning systems insuring that the release of new production lots does not overload the resources and that schedules are consistent with the steady flow of work-in-process according to targeted cycle times. It is not discussed, however, how targeted cycle times are calculated and if simulation is used to estimate performance measures of the manufacturing lines.

Production Management of Semiconductor Manufacturing

The importance of lead time reduction for productivity issues is widely acknowledged in semiconductor manufacturing. This judgement is supported by the numerous articles which have been published on the topic. Boebel [BR96], for example, discusses different effects of cycle time reduction focusing on yield increase which can be gained by faster passing through cycles of the learning curve. In the article findings of a project initiated to reduce cycle time at the Siemens/IBM DRAM production facility in Essonnes-Corbeil, France are presented. It was measured that the greatest part of the overall cycle times are caused by lacking machines or tool dedications followed by raw process time, waiting for operators, tool down time, transport, etc. The authors point out that the most important tool for quantifying targets and achievements is the line profile analysis.

The functional dependency between work-in-process, lead time, and line utilization (see figure 8.11) is also utilized by Martin [Mar98] who presents the notion of short cycle time manufacturing (SCM), i. e. using the X-factor¹ as a "more sensitive indicator of capacity problems than throughput, because the X-factor increases rapidly as the throughput approaches the effective capacity". The authors compare their approach with CFM² showing by means of a simple example that activities based on CFM would focus on a work center constraining throughput which would have very small effect on line performance (measured in X-factors). Next to measuring the X-factor at different work centers it is proposed that individual targets are set for these work centers which all together aggregate to an overall target X-factor. It is neither said how individual nor overall targets are to be derived; particularly no simulation methods are used to study the influence of variability on the X-factor.

Robinson et al [RGWAC99] discuss the validation of a simulation model of the Seagate read/write-head wafer production facility in Minneapolis, MN. To estimate lead times a model of the production line designed for long-term capital equipment needs has been simulated using discrete event simulation. The model had been extended and updated by industrial engineers in order to reflect former changes in the environment. This led to significantly shorter cycle times than observed in the actual production facility. A project has

¹ration between lead time and raw process cycle time

²Continuous Flow Manufacturing

therefore been set up to determine which factors were causing cycle time differences between the model and the actual factory and to add detail to the simulation model to bring cycle times closer to reality. The details which were added to the simulation model included actual start rates, actual tool quantities, current process flows, express lots, actual operator quantities, equipment dedication, and transportation. In the article the influence of each of these factors on the total lead time is discussed. The validation process could be partly supported by using data from the equipment tracking system: Comparing the number of available machines and the utilization showed that the predicted values were "quite similar" to the observed measures. Robinson et al mention that "the top cycle time tools in the resulting model also better matched the industrial engineers' perception of top cycle time contributors in the factory". A precise figure to further quantify the perception is not given, though.

Ming-Der Hu and Shi-Chung Chang present an approach using an analytical performance model to derive throughput rates and cycle time goals at work center level from overall production targets [HC00]. They make use of an analytical model based on approximate, decomposition methods (see [Whi83] [RK73], or [PA86]). The model is therefore quite similar to the one presented in chapter 3. It is limited though to $G/G/1$ nodes and a single product type. To validate the performance model it is compared to the outcome of discrete event simulation. Using the analytical model they derive the mean and squared coefficient of variation of the input processes at each work center from overall production targets. Explicitly the idea of using control charts for guiding the material flow through production lines is expressed. The target mean and control limits should be set from the model's performance measures. An implementation is not presented.

Bonal et al [B⁺01b] present a time series analysis of cycle times on work center level. A methodology called *Days Added* principally relying on an EWMA³ analysis of the average overall cycle time aggregated over products and process steps at a work center per week is presented. The main goal is a rapid detection of major changes in lead time. Bonal et al use the system to detect planned and unplanned bottlenecks. The deviation in the cycle time of work center k in week s with respect to its recent history is expressed by $DAAS_{k,s} = E_{k,s} - \bar{E}_{k,s}$, where $E_{k,s}$ is the moving average⁴ of

³Exponentially Weighted Moving Average

⁴The authors chose a weighting factor $\lambda = 0.4$

the aggregated cycle time of work center k in week s . $\bar{E}_{k,s}$ is the average of $E_{k,s}$ over the last 20 weeks, formally $\bar{E}_{k,s} = \left(\sum_{i=0}^{19} E_{k,s-i} \right) / 20$. To identify work centers driving changes in cycle time every week a pareto analysis of $DAAS_{k,s}$ is performed. As work centers having a higher number of process steps performed on them have a higher probability to appear in the first positions of the pareto analysis only work centers passing through the following filter are analyzed

$$E_{k,s} > \bar{E}_{k,s} + \sqrt{\frac{\sum_{i=0}^{19} (E_{k,s} - \bar{E}_{k,s})^2}{20}}. \quad (11.3)$$

The authors note that their approach would overcome the problems of other methodologies found in the literature⁵ which are solely based on targeted cycle time:

1. Long time to detect some cycle time trends
2. Overreaction to work centers with high variability

A reason for this is seen in the calculation of the target cycle times: "[the] targets of FF [the flow factor which is related to the X-factor] usually are, in some how, arbitrary".

Statistical Process Control

Statistical process control (SPC) is a widely accepted and practised methodology which is reflected in the large number of publications available on this subject. Most of the monographs concentrate either on the theoretical background or on the application of SPC in manufacturing environments (see Dietrich [DS98b], Amsden [A⁺91], Wheeler [WC90], Weihs [WJ99], Rinne [RM91], Pfohl [Pfo92], Horvarth [HU91], Bernecker [Ber90], VDI [CIM92, CIM91], Spenhoff [Spe91], or Uhlmann [Uhl82], for example).

Oakland [OF86, chapt. 14] discusses SPC in non-manufacturing environments: "data is data and whether the numbers relate to machine settings, process variables, prices, quantities, discounts, customers, or supply points is irrelevant: the techniques can always be used." The following applications are presented: Bank transactions times, profits on sales, forecasting income,

⁵they cited [Mar98] for example

employee absenteeism, errors in invoices, numbers of injuries that required first aid in a manufacturing plant, etc.

Rosander [Ros85] treats the use of SPC in various areas of the service industry including banking, insurance, government, health services, transportation, retail trade, business services, personal services, and public utilities.

Ryan [Rya89, p. 76] lists 61 references concerning applications of SPC in different scenarios. Only one of them studies control charts for queuing applications (see Lewellyn [Lew60]).

Statistical Inference of Queueing Systems

If performance measures of production lines are to be analyzed, methods of descriptive statistics and statistical inference have to be employed. The fundamental theories for the analysis of empirical data are described in numerous text books like Winkler [Win75], Lehoczky [Leh90], Sachs [Sac97], Fahrmeier et al [F⁺97], Hartung [Har93], Müller et al [Mül91], and Voß et al [Vos00a], just to name a few. Due to the special structure of queueing systems some publications deal specifically with statistical inference of queueing systems. Bhat and Rao [BR72] present various statistical methods focusing on estimation problems, model identification, and hypothesis testing. The article also contains an overview of the literature in this area.

Every discrete event simulation produces a vast amount of data — in most cases realizations of random variables. From the raw data performance measures of the model simulated have to be estimated. Since discrete event simulation is comparable to analysis by experimentation the usual procedures of experiment design apply. For proper analyses it is not sufficient to calculate point estimators, information about the precision of the estimators like confidence intervals have to be derived. In general, four critical problems encountered by all stochastic simulations can be identified (see [Fis73]):

1. estimation of the statistical reliability of sample performance measures, i. e. estimating the sample variance
2. need to extend the theory of distribution in order to facilitate the computation of confidence intervals
3. bias reduction by dilution of influence from the transient phase

4. variance reduction methods to accomplish a specified statistical reliability with less simulation time than pure random sampling

Literature dealing with simulation of queueing systems therefore discusses the use of statistical methods in order to tackle the mentioned problems of analyzing the output of a simulation run. In the scope of this chapter the main interest lies in the first two mentioned problems.

Lavenberg [Lav83a] and Payne [Pay88] present techniques to estimate transient and steady-state behavior from simulation output. The following four methods to estimate behavior in terms of the mean and the variance of performance measures are described in both monographs:

1. method of independent replications
2. method of batch means
3. method of regenerative states
4. spectrum and spectral density method

The third method is introduced among others by Fishman [Fis72, Fis73]. In particular Fishman [Fis72] derives point and interval estimates for 12 different performance measures including the number of parts in the system, waiting time, and lead time for multi-server queueing systems of type $GI/G/c$. All estimators are calculated by linear combination of three basic statistics

$$a_1 = \sum_{i=1}^{c-1} t_i, \quad (11.4)$$

$$a_2 = \sum_{i=c}^{\infty} t_i, \text{ and} \quad (11.5)$$

$$a_3 = \sum_{i=c}^{\infty} i t_i \quad (11.6)$$

using the sample sequence $\{t_i | i = 0, 1, \dots, \infty\}$ where t_i denotes the time spent in state i defined by i parts in the system.

In less detail these methods are also described by Gross and Harris [GH74]. In Heymann [DS90] Lehoczy [Leh90] discusses statistical methods used by Schmeiser [Sch90] who presents the four methods mentioned.

Schmeiser also focuses on input modeling, i. e. parameter estimation and distribution selection of simulation input parameters. This is also covered by Payne [Pay88].

The estimation of the variance of the sample mean is complicated by correlation which often exists between observations generated by simulation runs. This problem is covered in articles by Page [Pag63], Law [Law75], or Daley [Dal67]. Law and Kelton [LK84] compare five different methods (replication, batch means, autoregressive representation, spectrum analysis, and regenerative cycles) commonly used for the calculation of confidence intervals in case of autocorrelated time series. Besides presenting the methods mentioned confidence intervals are calculated for an $M/M/1$ system and a time-sharing computer model using the method of batch means, autoregressive representation, spectrum analysis, and regenerative cycles. It turns out that the method of batch means with 5 (or even fewer) batches performs nearly as well as any of the methods presented being the simplest and most inexpensive⁶ procedure. The only real competitor in terms of coverage is the spectrum analysis which is quite complicated and requires a larger computational effort because of the estimation of covariances. Interestingly Law and Kelton found the more complicated model of the time-sharing computer system to behave better statistically, making it more amenable to the presented statistical methods.

The spectral analysis is covered in the article by Fishman and Kiviat [FK67] which offers an introduction to the spectrum analysis of simulation-generated time series. The methodology is introduced and estimates of the correlogram and the spectral density are compared to their analytically derived values.

An advanced technique to construct confidence intervals based on spectral analysis is presented by Heidelberger and Welch [HW81]. The method estimates the logarithm of the averaged periodogram at zero frequency by using regression techniques.

Design and Control of Queues

According to Gross and Harris [GH74, p. 364] simulation models can be classified into two general types — *descriptive* and *prescriptive*. While the former type of models describe some situation, e. g. they are used to estimate

⁶concerning the complexity of calculating the confidence interval

or calculate performance measures of a specific situation, the latter type of model prescribes some type of action to be taken in order to direct the system into (maybe) optimal situations. Work on prescriptive models can be grouped into design models and models dealing with control of queues. Design models prescribe some configuration of parameters under which the model yields optimal performance measures in terms of a specified objective function, e.g. the optimal batch size at a work center to minimize work-in-process. Gross and Harris note that the literature available on design models is scarce. According to them even less literature is available for models dealing with control of queues. These models are not designed to specify optimal parameters but to detect changes in parameter settings. In general the authors classify design and control models into

1. rate-control policies which prescribe the way in which the arrival and/or service rates are to be changed in order to reach an optimum of some objective function,
2. models dealing with the optimization of queue discipline,
3. selection and optimization of scheduling rules like first come first served (FCFS), shortest-processing-time, most-imminent-due-date, etc., and
4. models detecting changes in parameters or derived performance measures.

For this work models of the fourth category are interesting. One of the few models of this type has been developed by Bhat and Rao [BR72]. They present a control chart approach to detect changes in the utilization of $M/G/1$ and $GI/M/1$ queueing systems.

11.1.4. Systems

Integrating shop-floor-control and simulation involves communication between various systems which should be briefly introduced in this section.

Shop-Floor-Control Systems

Shop-floor-control (SFC) systems, sometimes called manufacturing execution systems, collect data, execute manufacturing activities, and provide information immediately to supervisors on the manufacturing floor. These

systems are build on general process models to describe the manufacturing process flows, materials used, quality and test checks, resources planned, and work instructions and specifications needed (see [Spu92] or [Cam98]). By means of these models systems collect and organize critical data from the manufacturing floor in real-time.

To track and monitor manufacturing activities systems provide the following features:

- *Process modeling.* Includes definition and control of operations, specifications, parameters, instructions, and alternate process paths.
- *Work-in-process tracking.* Tracks work-in-process by lot, batch, or unit/serial number. Provides a complete history of production, traceability, and genealogy of materials.
- *Specification/Documentation control.* Provides secured control of specifications, instructions, and process data tolerances. Includes interfaces to word processors and graphics systems.
- *Material control.* Defines all product relationships including bills of material, options, alternate process paths. Allocates material and controls its usage.
- *Resource management.* Tracks all nonmaterial resources used in production including equipment, tools, buildings, and operators. Provides equipment history, preventive maintenance, and a collection of repair, labor, parts, and calibration data.
- *Real-time production monitor.* Provides on-line snapshots of current throughput and yield results. Provides historical views of archived information.
- *Container tracking.* Provides the ability to track transport lots by containers. Maps containers to lots for automated environments.

Besides these basic features often additional modules for quality management, SPC applications, cost accounting, and lot scheduling are available. The Mainz wafer line, for example, uses the shop-floor-control system ME-SA™ by Camstar Systems.

Statistical Software Packages

Today various software systems for statistical analyses are available. Systems like SPSS, SAS, MatLab, Minitab have their strength in the analysis of large amounts of data and feature the possibility of connecting to databases. Other systems like Mathematica, Maple, or spreadsheets like MS Excel or Lotus 1-2-3 provide statistical methods but are not targeted for the amount of data needed for the analyses and applications presented.

SAS which has been used for the preparation of this chapter is a family of applications which started out as a package for statistical analysis in 1967. Today the statistical core components have been extended by applications ranging from spreadsheets, business graphics, geographic information systems, C-compilers, tools for building WWW interfaces, etc. The software is developed and distributed by the SAS Institute.

As the description of the SAS system is beyond the scope of this work the interested reader should refer to one of the many publications on the system, see Delwiche and Slaughter [DS98a], Dufner, Jensen, and Schumacher [DJS92], Falk, Becker, and Mahron [FBM95], Gogolok, Schuemer, and Ströhlein [GSS92], Batz [Bat94], Kähler and Schulte [KS90], Göttsche [Göt90], Nagl [Nag92], or Aronson [AA90], for example.

Integration

Figure 11.1 shows an UML deployment diagram focusing on the systems needed to integrate simulation and shop-floor-control system. Every work center in the production line is equipped with a PC that is used inter alia for claim transactions. The statistical packages SAS is used to access actual data from the shop-floor-control system and also plan parameters from the EPOS database. Reports and charts which are created by SAS can be pushed onto an HTTP server making results available to all authorized users in the company via the intranet.

For some applications like the work-in-process forecast presented in section 11.3 the model generator needs to initialize the simulation model with data gathered from the shop-floor-control system. This can be realized by the JDBC connection between the model generator and the database of the manufacturing execution system.

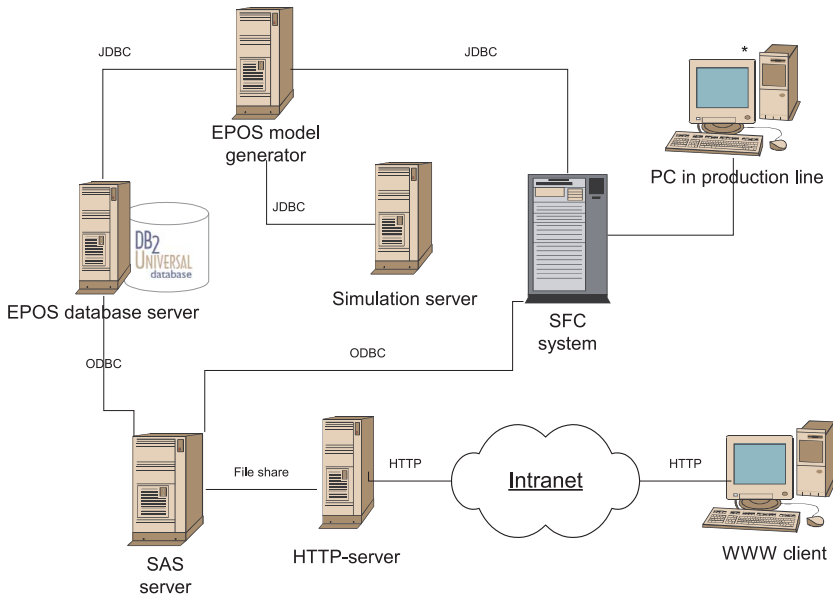


Figure 11.1.: Integration of simulation, shop-floor-control system, and the statistical package SAS.

11.1.5. Analysis of Stochastic Processes

As described in the literature review (see 11.1.3) there is a theory available for estimating performance measures of queueing models from simulation output. It has to be noted, though, that while the analysis of simulation runs and the study of empirical data from production lines bear many equalities, there exist some important differences concerning the amount of influence the observer possesses. It is quite obvious, for example, that the method of

Comment: This chapter contains several statistics computed from real data taken from the shop-floor-control system of the wafer production line of the IBM Deutschland Speichersysteme GmbH in Mainz. As part of the data is classified confidential the quantities of results like batch sizes, waiting times, work-in-process, etc. are removed from the charts presented.

Field	Type	Description
Wafer	integer	the ID of each wafer
Op	string	the operation ID
Command	string	the specific command of the transaction
Movetype	string	the type of command, e. g. move-in or move-out
Timestamp	timestamp	the time the transaction was processed
ARC	string	archive status of the transaction
Employee	string	employee who initiated the transaction
Batch	integer	the batch the wafer is part of
ResourceId	string	ID of the machine at which the transaction occurred
Defect	string	the code of the defect if any occurred
HT	string	head type (product)
CTRLNum	integer	running transaction number

Table 11.1.: Structure of the transaction table

independent replications (see 11.2.3) is not applicable when it comes to the analysis of real production lines. Moreover, while the use of certain estimators seems attractive when analyzing simulation output the same estimators are rather difficult to use when analyzing data from shop-floor-control systems. It is therefore important to discuss the structure of data available and the type of logistical processes found in the specific production line being analyzed.

In section 11.2 quality control charts for the waiting time in front of a work center and the process time observed at a process step are constructed. These two processes and their behavior in terms of distribution, stationarity, and correlation are analyzed up front. Observations of neither process time nor waiting time are directly available from the transaction table presented but can be calculated from it. This is presented in section 11.2.2.

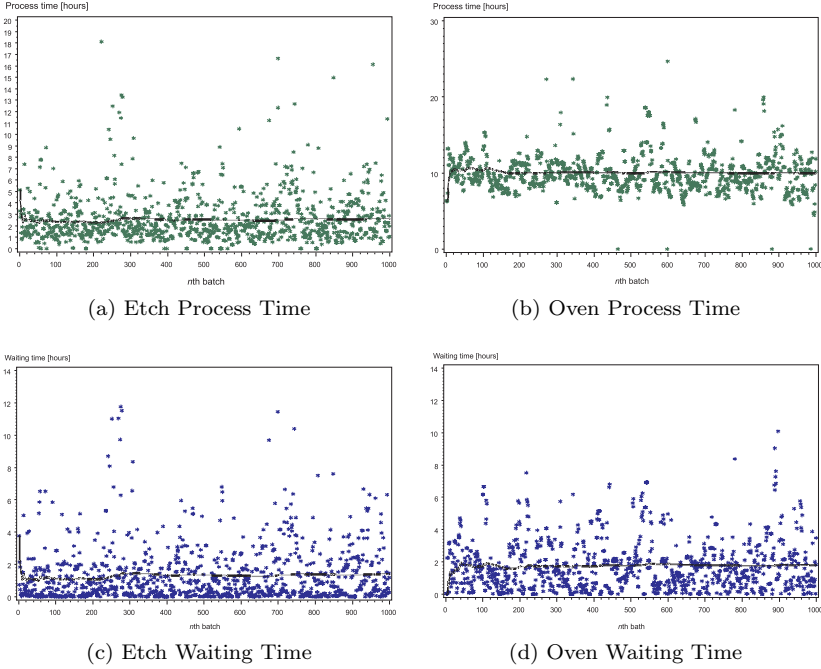


Figure 11.2.: Process and waiting time at two different machines. Figure (a) and (b) show the process time at an etch machine and a substrate anneal oven, respectively. Each chart shows 1000 observations of batches and the average over time. Figure (c) and (d) present waiting times at the same machines of the same batches.

Structure of Available Information

Table 11.1 shows the typical structure of the transaction table of shop-floor-control systems⁷. The source for the data stored are claim transactions initiated by line operators:

- Before the processing of any part at any operation a part has to be

⁷The table is presented in a non-normalized format to facilitate the discussion.

claimed into operation. To do this the operator enters the part number and the number of the operation into the shop-floor-control system. The system checks whether this process step is valid and then permits or rejects processing of the part, respectively.

- After a process has been finished each part is claimed out of operation. The occurrence of any possible defect or reason for rework is entered at this step. The part is then virtually moved into the queue of the following operation.

The shop-floor-control system records all information from these claim transactions, e. g. the system stores the batch number operation ID and time of the transaction in its database. The structure of table 11.1 is transaction-oriented, i. e. to be able to obtain the process time, for example, one has to transform the data (see section 11.2.2).

This transaction table is the only source of information used by analyses and applications presented in this work. This has the following three reasons: (i) the data is very generic and permits the derivation of various statistics, (ii) other information like parts which have been processed together in production processes is generally not available, and (iii) the structure of the transaction is basic to every shop-floor-control system, thus the methods presented in this work can be applied in other situation, as well.

Stationarity

The procedures developed for the analysis of simulation generated output can often be applied to the analysis of data taken from real production systems. In simulation environments one is often confronted with two distinct phases during a simulation run — the transient and the stationary phase. As the studied production line has already been in operation for many years it is assumed that no transient phase exists.

Comment: In this chapter the time series x_1, \dots, x_n is thought of a finite realization of a stochastic process $(X_t)_{t \in T}$. In general the index set T is equal to \mathbb{N} for the applications subsequently studied, i. e. the process X_1, X_2, \dots is a sequence of (usually) correlated random variables. The usual notation with X_i as a random variable and x_i as one of its possible realizations is applied.

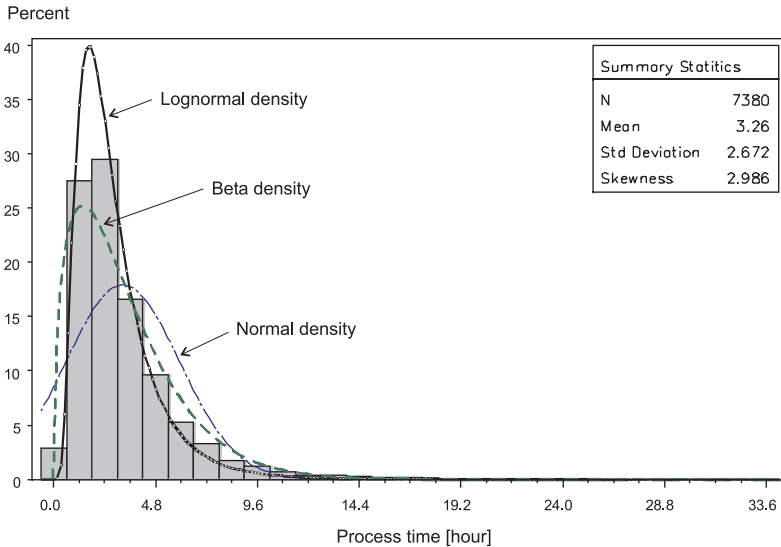


Figure 11.3.: Histogram of process times at an etch machine and fitted normal, lognormal and beta distributions

To verify this assumption the time series x_i for $i = 1, \dots, n$ from either process time and waiting time have been plotted in figure 11.2. The time average $\bar{x}_i = i^{-1} \sum_{j=1}^i x_j$ for $i = 1, \dots, n$ is calculated for every observation in the series. The charts show that no transient phase in neither the process time, nor the waiting time is apparent.

The analysis has been carried out for about 40 of the most important work centers for a time horizon of nine months. For the process time in nearly all cases the same result was found (deviating behaviors could be explained by introduction of different manufacturing processes). As the waiting time depends on the utilization of the work center changing averages could be observed, but often even the waiting time distribution has been stationary, because production control normally tries to run the manufacturing facilities on a specific level of utilization.

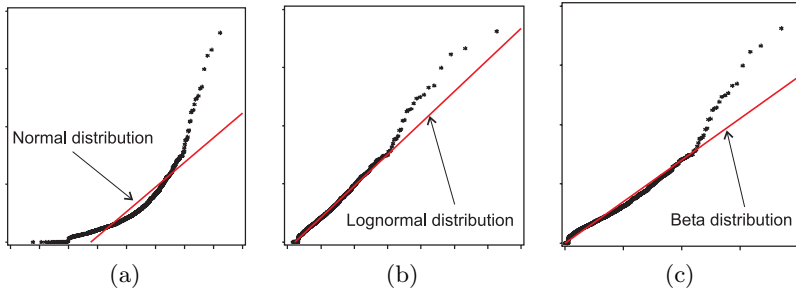


Figure 11.4.: Quantile-Quantile Plots for normal, lognormal, and beta distributions

Process and Waiting Time Distributions

One is interested in the type of distribution of the process and waiting time as control charts of Shewhart type assume normally distributed processes. The Kolmogoroff-Smirnov goodness of fit test reveals very low p-values (below 0.001) for both process. The null-hypothesis that either process is normally distributed can therefore be rejected.

Different distributions have been fitted on the empirical data. Figure 11.3 shows a histogram of the process time with fitted normal, lognormal, and beta distributions. The lognormal distribution can be shown to produce the best fits. Also the Weibull, gamma, and exponential distributions were analyzed, but showed inferior results.

To gain further insight Q-Q plots comparing quantiles of the theoretical and empirical distribution function have been constructed, see figure 11.4. One can see that the symmetrical normal distribution fails as a model due to the skewness of the observed process. The skewness in the process time can easily be explained: There is a lower bound on the time many production processes last, some machines even control the process time in milliseconds. There are many possible causes for larger process times, though, e.g. down times like planned or unplanned machine failures, breaks, lunches, set-up times, etc. Moreover, there is a substantial amount of operator handling time before and after the physical process.

The lognormal and beta distributions perform better in modeling skewness. Both models differ in larger process times. Here, the question arises in how far these large values are really caused by the underlying process

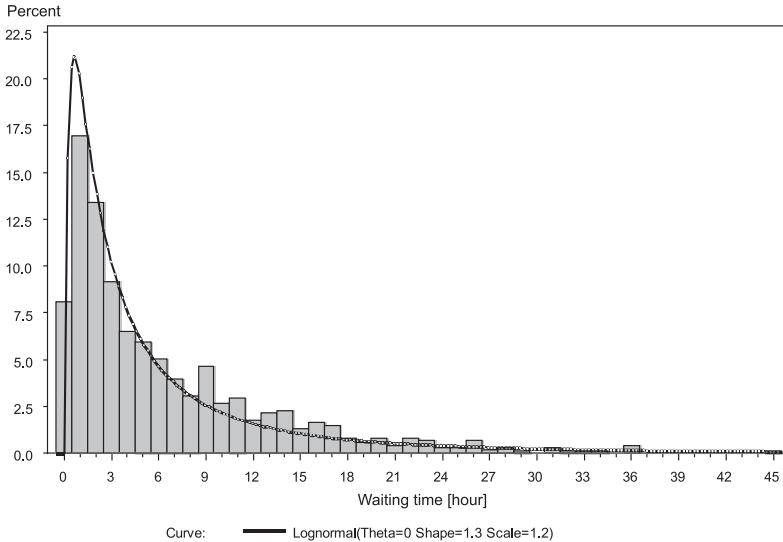


Figure 11.5.: Waiting time at a sputter work center and fitted lognormal density curve

or rather by measurement problems, technical, or even political decisions influencing the manufacturing process. Analyzing causes for the very large values would require much effort and has not yet been carried out.

In figure 11.5 a histogram of the waiting time including a fitted lognormal distribution is presented. The lognormal model produced the best fits for this process; for other waiting time processes, the exponential distribution produces better fits, since in high traffic cases the waiting time is approximately exponentially distributed (see [GH74, p. 323]).

Inter-Arrival Processes

The analysis of inter-arrival times throughout the whole production line yields some interesting results. Figure 11.6 shows the estimated coefficient of variation (CV) in percent for each operation of the production line ordered by the flow of material from start of production to the last operation. The line indicates the trend of the CV. High variation can be observed especially

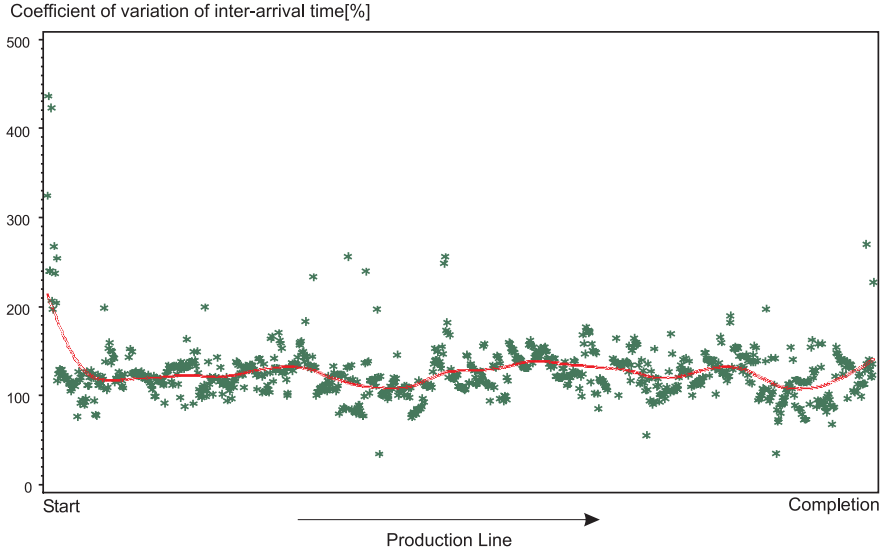
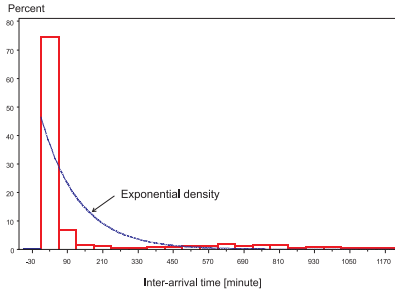


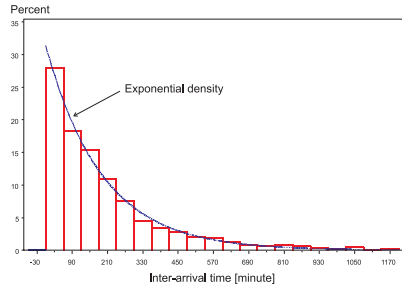
Figure 11.6.: Coefficient of variation (CV) of inter-arrival times over the manufacturing processes from the start of production to completion. Each star represents an estimate of the coefficient of variation at each operation.

at the beginning of the production. Analyzing the start of build (SOB) characteristics showed that parts are started just once per shift. This is done as one of the very first operations is performed at an oven having a very large batch size and process time. Parts are therefore scheduled corresponding to this first bake process. This implies that a *bubble* of parts moves through the first operations causing the high variation: inter-arrival times at the first operations are characterized by very short times between the arrival of many batches and then a long time with no arrivals as machines behind the oven await the end of the bake process.

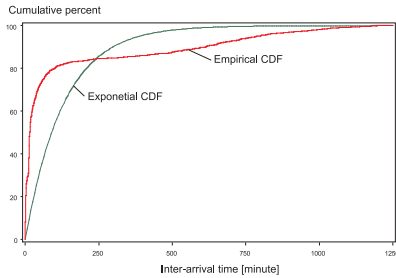
Moreover, it can be noted that throughout the manufacturing line the coefficient of variation is close to 100%, an effect which has been also been noted by Connors et. al. [CFY96]. This leads to the hypothesis that the arrival process could be a Poisson process. Figure 11.7 investigates this hypothesis by showing charts from two different operations, on the left side,



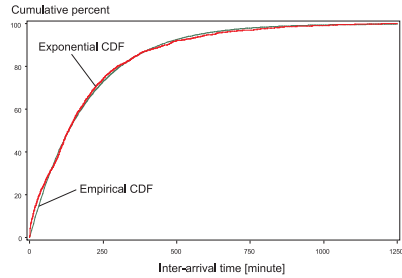
(a) Histogram



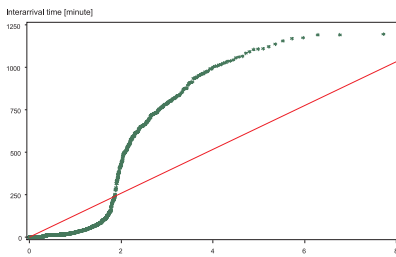
(b) Histogram



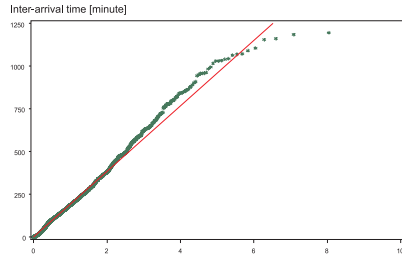
(c) Distribution function



(d) Distribution function



(e) Q-Q Plot



Q-Q Plot(f)

Figure 11.7.: Comparison of inter-arrival times at different processes. The charts on the left are generated from inter-arrival times at the first bake process, the charts on the right side stem from a plating process in the middle of the process flow.

(a), (c), and (e) stem from the first bake operation, charts on the right side, (b), (d), and (f) are taken from a plating operation in the middle of the process flow. Each chart compares empirical data with a fitted exponential distribution, in the first row the histogram is plotted against the exponential density curve, in the second row the cumulative density function (CDF) is shown, and the third row contains two exponential Q-Q-plots. The exponential model obviously is not the right choice to model the arrival process of the first bake operation. The operation in the middle of the process flow can be described by it quite well.

The theoretical substantiations for the observed effect can be found in a theorem about the superposition of sparse renewal processes which is given in the following.

Let $N_{n,i}(t)$ for $n = 1, 2, \dots, \infty$ and for each $i = 1, 2, \dots, k_n$ where $k_n \rightarrow \infty$ as $n \rightarrow \infty$ be renewal counting processes which by this definition form a triangular array of stochastic processes. The inter-arrival time distribution of these renewal processes are denoted by $F_{n,i}(t)$. For every n the processes

$$N_{n,1}(t), N_{n,2}(t), \dots, N_{n,k_n}(t) \quad (11.7)$$

are assumed to be independent. The superposition process $N_n(t)$ is then defined by

$$N_n(t) = \sum_{i=1}^{k_n} N_{n,i}(t) \quad \text{for } t \geq 0. \quad (11.8)$$

If all processes $N_{n,i}(t)$ are poisson processes the superposition process is itself a Poisson process, but this is not the case for general processes. It can be shown that these processes are not even renewal processes as the inter-arrival times are neither independent nor identically distributed. The following definition specifies what is meant by sparse renewal processes.

Definition 11.1.1 *The triangular array of processes $N_{n,i}(t)$ is called infinitesimal if for every $t \geq 0$*

$$\lim_{n \rightarrow \infty} \max_{1 \leq i \leq k_n} F_{n,i}(t) = 0. \quad (11.9)$$

A rationale for the Poisson assumption made in various situations, especially in queueing systems, is given by the subsequent theorem. From that point of view the result is quite comparable in its significance to the central limit theorem which serves as a foundation for the widespread use of the normal distribution.

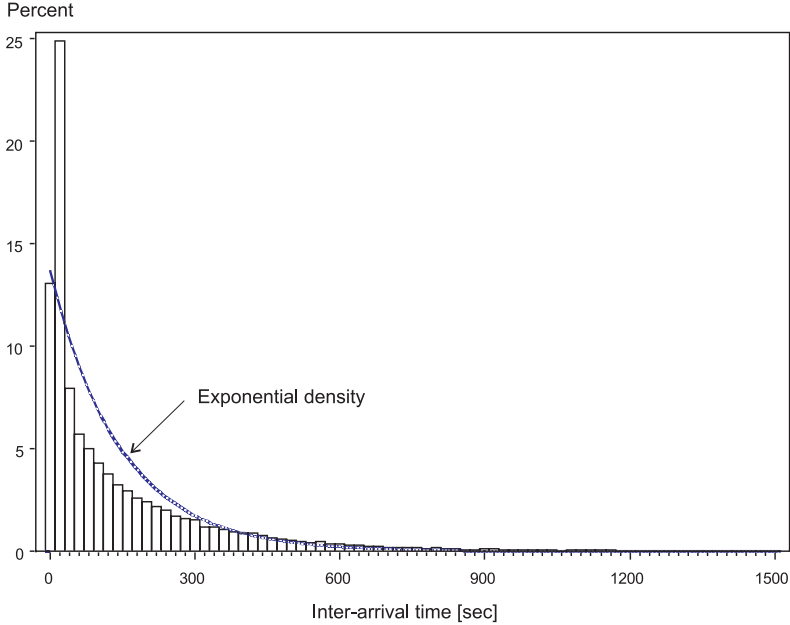


Figure 11.8.: Histogram of inter-arrival times at a photo cluster

Theorem 11.1.1 (Superposition of renewal processes) *Let $N_{n,i}(t)$ be an infinitesimal array of renewal processes with superposition $N_n(t)$. If and only if*

$$\lim_{n \rightarrow \infty} \sum_{i=1}^{k_n} F_{n,i}(t) = \lambda t \quad (11.10)$$

then

$$\lim_{n \rightarrow \infty} P \{N_n(t) = j\} = \frac{e^{-\lambda t} (\lambda t)^j}{j!}, \quad \text{for } j = 0, 1, 2, \dots \quad (11.11)$$

See [KT75, p. 223] for the proof.

This result suggests to analyze machines where the input stream is formed as a superposition of many streams. This is, for example, often the case

in semiconductor industries which can be characterized by their highly re-entrant manufacturing structures, i. e. because of the very capital-intensive production facilities many machines have to be used for several process steps.

Figure 11.8 shows a histogram of the batched inter-arrival time at a photo cluster which is used for two operations in every photo process, i. e. twice per layer. A fitted exponential curve reveals the differences between the theoretical model and the observations. While the fit is acceptable for larger inter-arrival times there are too many very short intervals compared to the model. This can be explained by the way operators perform the claim procedures. Often they perform several in-claims at the shop-floor-control system, process the parts of all claimed batches, and finally perform the out-claim. This results in too many short intervals.

Observing Little's Law

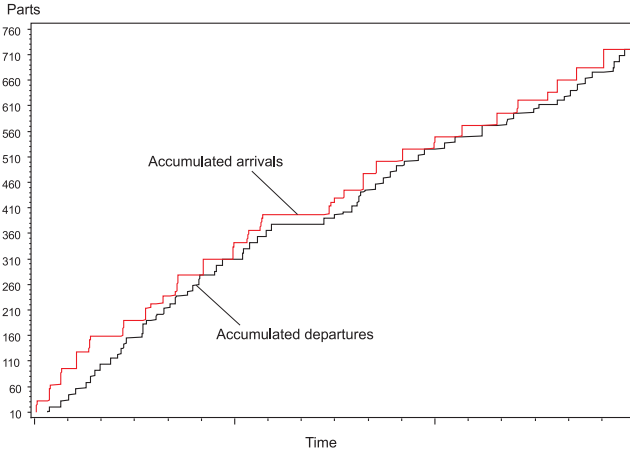
The relationship between work-in-process and lead time is described by Little's law, see theorem 3.1.1 on page 76. This famous result from queueing theory can be written as

$$L = \lambda W \quad (11.12)$$

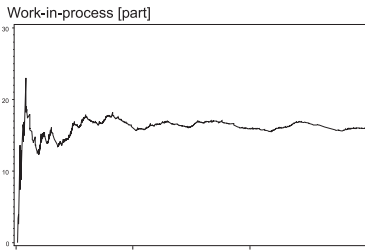
with $L = E[N]$ as the expected number of units in a queueing system, $W = E[Q]$ as the expected time spent by a unit in the system (lead time), and $1/\lambda = E[I]$ as the expected time between two consecutive arrivals to the system.

The theorem which is widely used in the calculation of performance measures of queueing systems can be seen from the transaction data of shop-floor-control systems. Figure 11.9 shows charts that visualize the processes involved. The data which is taken from an arbitrary work center in the wafer manufacturing line can be characterized as follows: With $0 = t_0 < t_1 < \dots < t_k$ as $k+1$ points at which the system state has changed, let the time series p_{t_0}, \dots, p_{t_k} denote the work-in-process and a_{t_0}, \dots, a_{t_k} and d_{t_0}, \dots, d_{t_k} the number of parts which have arrived or departed until time t_i , respectively. The lead times of m units that departed from the system are denoted by w_1, \dots, w_m .

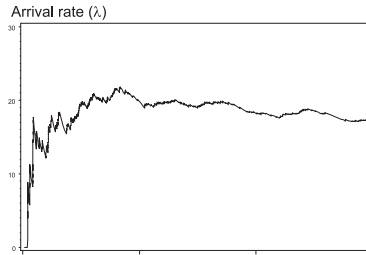
Hence, the time averages $L(t_i)$ (work-in-process), $\lambda(t_i)$ (arrival rate), and



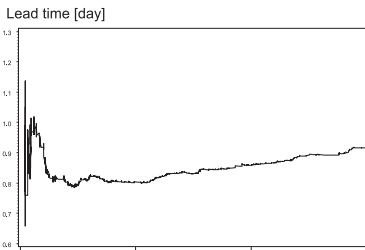
(a) Accumulated arrivals and departures, a_{t_i} and d_{t_i}



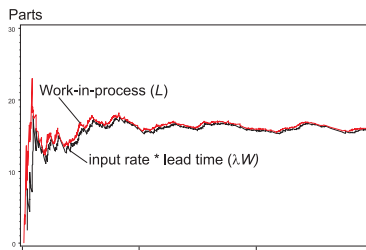
(b) Work-in-process $L(t_i)$



(c) Arrival rate $\lambda(t_i)$



(d) Lead time $W(t_i)$



(e) $L(t_i)$ and $\lambda(t_i)W(t_i)$

Figure 11.9.: Relationship between work-in-process and lead time

$W(t_i)$ (lead time) at times t_i are defined by

$$L(t_i) = \frac{1}{t_i} \sum_{j=1}^i (t_j - t_{j-1}) p_{t_j}, \quad (11.13)$$

$$\lambda(t_i) = \frac{a_{t_i}}{t_i}, \quad \text{and} \quad (11.14)$$

$$W(t_i) = \frac{1}{d_{t_i}} \sum_{j=1}^{d_{t_i}} w_j \quad \text{for } i = 1, \dots, k. \quad (11.15)$$

The first chart, figure 11.9.a, shows the accumulated arrivals a_{t_i} and departures d_{t_i} at the work center. At each point in time the difference $a_{t_i} - d_{t_i}$ is equal to the current work-in-process in the system⁸. The horizontal distance between the two lines represents the lead time in the system, compare [Eil69]. The subsequent charts, (b)-(d) show the time averages $L(t_i)$, $\lambda(t_i)$, and $W(t_i)$. Figure 11.9.e presents the combination of the former processes, i.e. one line represents the work-in-process $L(t_i)$ and is therefore equal to 11.9.b, the other one represents the product of $\lambda(t_i)$ and $W(t_i)$. With increasing t the difference between $L(t)$ and $\lambda(t)W(t)$ apparently decreases

Autocorrelation

Statistical analysis of the logistical processes in production lines are complicated due to the existence of autocorrelation, e.g. waiting times of consecutive customers are likely to be correlated, i.e. the probability that the $(n+1)$ th part experiences a long waiting time is high knowing that the n th parts experienced a long waiting time. Thus, the random variables of the waiting process are not independent, hence standard statistical procedures designed for independent observations cannot be applied directly. Figure 11.10 shows the time that parts have to wait in front of an etch machine to be processed. The sample has been taken from the shop-floor-control system. One can already see from this figure that the waiting time of successive observations is positively correlated.

While positive autocorrelation of output processes like waiting time, queue length, number of parts in the system, etc. is quite obvious it is also present in input processes like service or completion times. This is not

⁸Assuming the accumulation is started with an empty system

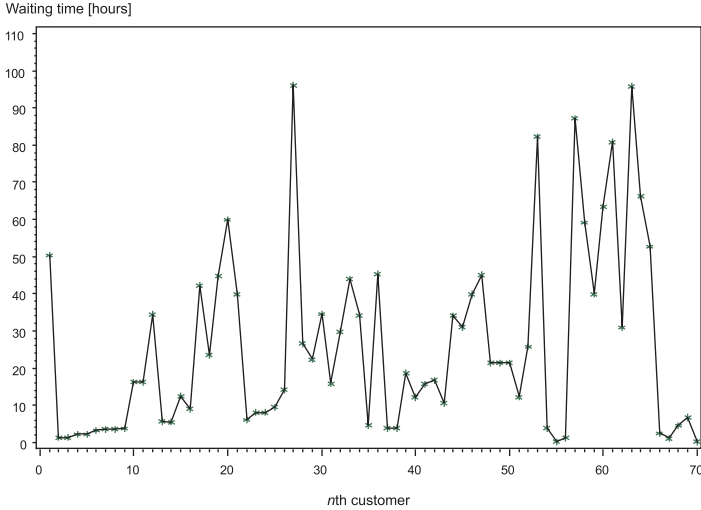


Figure 11.10.: Waiting time of the n th part in front of one of the bottleneck machines

as apparent as it is for output processes. In case of a sputter machine with computer-controlled process times, for example, neglecting target wear out it is not directly obvious why consecutive runs should be correlated. In fact, consecutive process time might not be correlated at all, autocorrelation is rather introduced by batch processing and the way the shop-floor-control system reports process times. If the transport batch size is small compared to the machine's process batch size, then several transport batches have to be collected to create one process batch. Thus, consecutive transport batches are processed in same run and experience the same process time.

To specify the autocorrelation more precisely some background from time series analysis is needed. Let x_1, \dots, x_n be a sequence of observations, e. g. let x_i denote the waiting time of the i th part. The autocovariance is a measure of the relation between observations of the time series. The autocovariance of lag 1 is therefore a measure of relation between x_t and x_{t+1} for $t = 1, \dots, n - 1$. The covariance between two random variables X and Y is defined by

$$\text{Cov}[X, Y] = \text{Cov}[Y, X] = E[(X - E[X])(Y - E[Y])]. \quad (11.16)$$

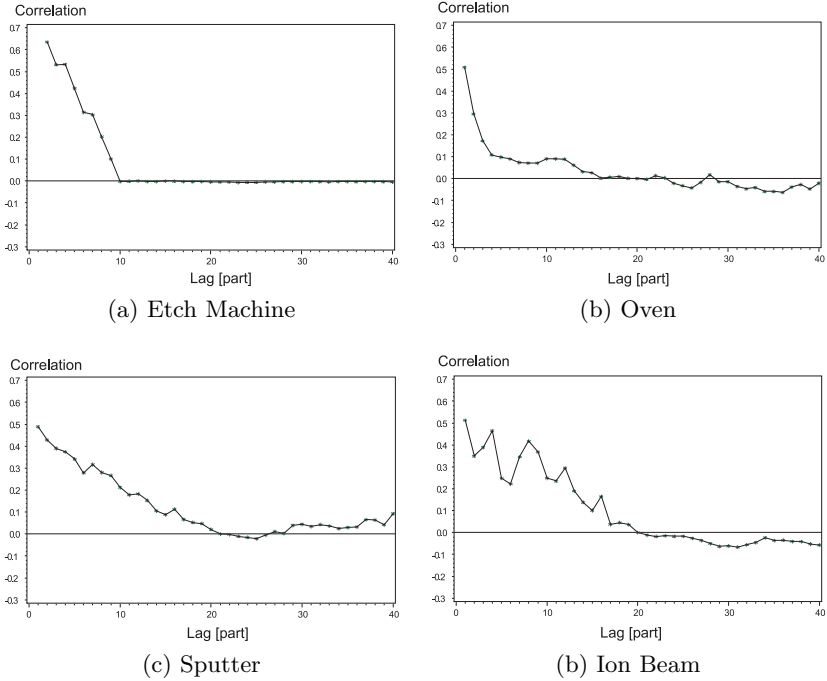


Figure 11.11.: Autocorrelation $r(k)$ of lag k , $k = 1, 2, \dots, 40$ of the waiting time at four different machines

As x_t and x_{t+k} with lag k can be thought of as realizations of two random variables an estimator for the autocovariance $c(k)$ of lag k of the time series is

$$\hat{\gamma}(k) = \hat{\gamma}(-k) = \frac{1}{n} \sum_{t=1}^{n-k} (x_t - \bar{x})(x_{t+k} - \bar{x}) \quad (11.17)$$

with \bar{x} as the mean of the time series (see [Har93]). The autocorrelation of a time series is a normalization of the autocovariance, an estimator is given by

$$\hat{\rho}(k) = \frac{\hat{\gamma}(k)}{\hat{\gamma}(0)} = \frac{\sum_{t=1}^{n-k} (x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^n (x_t - \bar{x})^2} = \hat{\rho}(-k). \quad (11.18)$$

To estimate the autocorrelation using (11.18) the process X_1, \dots, X_n is

assumed to be covariance stationary, formally $\text{Cov}[X_t, X_{t+k}] = c(k)$ which means that the covariance between any two random variables of the sequence can be expressed as the covariance function $c(k)$ which solely depends on lag k and not on the position t within the sequence.

The autocorrelation function $\rho(k)$ is the correlation between two members of the sequence with a separation of k units. It has the following properties:

$$\rho(0) = 1, \quad (11.19)$$

$$\rho(k) = \rho(-k), \quad (11.20)$$

$$-1 \leq \rho(k) \leq 1. \quad (11.21)$$

Using formula (11.18) the autocorrelation for $k = 1, \dots, 40$ has been estimated for every machine of the production line. Figure 11.11 shows the so-called *correlogram* for different machines. One can see that nearby waiting times are clearly positively correlated and that correlation decreases with increasing distance, formally $\lim_{|k| \rightarrow \infty} \rho(k) = 0$. When estimating autocorrelation of lag k the sample size n has to be large compared to k . Welch [Lav83b] proposes $n \geq 4k$, i.e. for a maximum lag of 60 at least 240 observation should be present which has not been a problem as twelve month of wafer production have been used to create the charts in figure 11.11.

The presence of positive correlation tends to reduce the amount of information gained per observation. The value of x_{t+1} knowing x_t provides little information in case the corresponding random variables are correlated. This becomes especially apparent when estimating the sample variance $\text{Var}[\bar{X}]$ which can be done in case of independent samples by

$$\text{Var}[\bar{X}] = \frac{\sigma^2}{n}. \quad (11.22)$$

This equality does not hold in case of autocorrelation, though. Welch [Lav83b] derives the correct sample variance with non-zero autocorrelation:

$$\text{Var}[\bar{X}] = \text{Var} \left[\frac{1}{n} \sum_{t=1}^n X_t \right] = \frac{1}{n^2} \text{Var} \left[\sum_{t=1}^n X_t \right] \quad (11.23)$$

$$= \frac{1}{n^2} \left(\sum_{t=1}^n \text{Var}[X_t] + 2 \sum_{k=1}^{n-1} \sum_{j=k+1}^n \text{Cov}[X_k, X_j] \right) \quad (11.24)$$

Using the symmetry of the covariance function the result can be written as

$$\text{Var}[\bar{X}] = \frac{1}{n^2} \left(n\gamma(0) + 2 \sum_{k=1}^{n-1} (n-k)\gamma(k) \right). \quad (11.25)$$

This can further be simplified by using $\gamma(k) = \gamma(-k)$ and $\rho(k) = \gamma(k)/\gamma(0)$

$$= \frac{1}{n^2} \sum_{k=-(n-1)}^{n-1} (n-|k|)\gamma(k) \quad (11.26)$$

$$= \frac{\gamma(0)}{n} \sum_{k=-(n-1)}^{n-1} \frac{n-|k|}{n} \rho(k) \quad (11.27)$$

$$= \frac{\sigma^2}{n} \sum_{k=-(n-1)}^{n-1} \frac{n-|k|}{n} \rho(k). \quad (11.28)$$

As $\rho(k)$ goes to zero when k gets large an approximation of equation (11.28) for large n is given by

$$\text{Var}[\bar{X}] \approx \frac{\sigma^2}{n} \sum_{k=-\infty}^{\infty} \rho(k) \quad (11.29)$$

$$= \frac{1}{n} \sum_{k=-\infty}^{\infty} \gamma(k). \quad (11.30)$$

Conway [Con63] presents an example showing the effect of ignoring autocorrelation. A job shop is simulated and the average time a job spends in the system is estimated from a sample of 100 jobs. The variance of the sample mean is computed as $\text{Var}[\bar{X}] = \sigma^2/100$ assuming the lead times were i.i.d.⁹ and also by the method of batch means. Comparing the standard errors of the mean the method of batch means yielded a 2.89 times larger standard error than the method ignoring autocorrelation.

This figure can be compared to the results of an analysis carried out on the waiting time at several machines in the wafer line, see table 11.2. Using the shop-floor-control system waiting times for batches at different work centers have been calculated. To estimate the variance of the sample

⁹independent, identically distributed

Work center	N	$\sqrt{\sigma^2/N}$	B.M.	Factor
Sputter	12719	0.3015	0.7042	2.3351
Tester	384	0.0671	0.1102	1.6408
Sputter # 5 & 7	2341	0.3114	0.5403	1.7346
Sputter # 6	689	0.5112	1.8940	3.7048
Sputter UC/OC	1258	0.2288	0.4514	1.9732
Etch machine	7394	0.1110	0.3153	2.8387
Insulation bake	489	0.3592	0.7861	2.1880
GMR oven	592	0.9086	2.1483	2.3643
Ion beam sputter 4	372	2.5832	7.6320	2.9544
Ion beam sputter 5	400	3.1081	8.3409	2.6835
Manual microscope	32708	0.0676	0.0946	1.3989
Automatic microscope	1600	0.1287	0.1251	0.9714
Automatic visual test	2374	0.3698	0.6506	1.7592
Photo cluster	4262	1.1402	2.0505	1.7982
Plating cell	7890	0.0070	0.0072	1.0294
Stepper	16228	0.1239	0.3330	2.6863
Anneal oven	1287	0.0523	0.0755	1.4449
Profiler	2856	0.0390	0.0472	1.2105
Ion mill	6937	0.1927	0.3009	1.5616
Sputter A	1259	0.5695	2.2438	3.9397
Sputter C	890	0.1244	0.2210	1.7757
Sputter C	14999	0.0429	0.1294	3.0124

Table 11.2.: Comparison between method of batch means (B.M.) and ignoring autocorrelation. For each machine the sample size, the mean standard error ignoring autocorrelation $\sqrt{\sigma^2/n}$, the mean standard error calculated by the method of batch means, and the factor comparing these two values is shown.

mean $\text{Var}[\bar{X}]$ (or the mean standard error $\sqrt{\text{Var}[\bar{X}]}$) the method assuming i.i.d. random variables and the method of batch means were compared. The result is expressed by the factor in the last column of table 11.2. These factors can be compared to the observation made by Conway.

11.2. Logistical Process Control (LPC)

Logistical process control (LPC) is the application of methods from statistical process control (SPC) to logistical processes. Differences between logistical and physical processes in terms of statistical behavior, correlation structure, derivation of targets, etc. demand the treatment of LPC on its own.

11.2.1. Overview

Schwinn [Sch93, p. 90] notes that problem determination is strongly aided by control of plans already being realized. New plans are triggered when differences between a planned target and a realized actual value are observed. Progress control which uncovers predictable plan/actual discrepancies by the use of early warning systems which present possible risks to users in advance is standard in medicine and military but has just in recent times been introduced into economy.

In mass production early warning systems for logistical measures are rather sparse. An exception is the approach by Bonal et al [B⁺01b] (see section 11.1.3). The system presented has the disadvantage that it does not compare actual measures to planned targets and therefore is not capable of determining the possible potential of logistical processes. SPC on the other hand is an early warning system that uses targets. In [fQ90, p.19] attributes that are controlled by SPC are defined as product characteristics like length, weight, etc. or process characteristics like temperature, pressure, or piston stroke. Logistical attributes are not mentioned. One page later [fQ90, p.20] the requirements for SPC are summarized:

1. Quality requirements demanded of the product have to be known.
2. Attributes that characterize product quality and process yield should be measured.
3. A prerequisite is the possibility of controlling the process.

The first requirement, i.e. specifying a target value for a process is rather straightforward for SPC but creates real difficulties for LPC. Only by means of simulation or analytical methods it becomes possible to specify feasible targets on different levels of aggregations, e.g. the ability to determine a lead time target for a work center at a specific utilization level makes a quality

control chart for this performance measure possible. Thus, a system for *integrated simulation* is the prerequisite on the plan side and a shop-floor-control system on the actual side.

Classification of Plan/Actual Comparisons

Before the actual design of the logistical quality control charts is presented an overview of characteristics of plan/actual comparisons should help to classify the control charts. In general, the following properties of plan/actual comparisons can be identified:

- *Level of aggregation.* The values being compared can be on any level of aggregation ranging from very specific to very general, e. g. the waiting time of a product and work center specific operation compared to the overall waiting time of the whole production facility.
- *Type of information.* If a model is used to derive the plan values the figures to be compared can either be input parameters or output values of the model. The process time, for example is an input parameter of the simulation model, whereas the waiting time is a calculated output performance measure.
- *Amount of comparisons.* For each parameter a set of objects exists where it can be measured or planned, e. g. the lead time can be measured and predicted at any work center of the production line. Plan/actual comparisons can include all the available comparisons or focus on specific subsets. A pareto analysis, for example, could be employed to concentrate only on the top ten lead time contributors.
- *Planning horizon.* The parameter to be analyzed can be of different nature depending on the purpose and the horizon of the planning task. This might reach from the comparison of the planned and estimated due date of an order up to the long-term planned utilization of the production line. A classification into operational, tactical, and strategic can be useful. Generally the planning horizon determines the amount of actual data needed to calculate the comparison.
- *Repetition.* Comparisons can be created just once or on a repeated basis. If the latter is the case, the interval between creation is of interest. This is clearly determined by the purpose of the comparison.

Goals of Logistical Process Control

A quality control chart is a plan/actual comparison. The last section has shown that many aspects influence the creation of plan/actual comparisons and thus quality control charts. It is therefore important to keep in mind the purpose of the charts presented here¹⁰. The logistical process control charts presented in this chapter have the following goals:

- *Decision support system.* Logistical process control is a decision support system that helps detecting problems in logistical processes found in manufacturing. Compared to *traditional* SPC which states explicitly what actions are to be taken in case of out-of-control samples, LPC focuses more on the analysis of differences than the direct initiations of actions. This has to do with the complex structure of the involved processes: An increase in waiting time at a specific work center can have many different causes which can well lay outside the scope of the work center.
- *Easy interpretation.* Resulting charts and reports should be easy to interpret and accessible to a broad range of manufacturing personnel ranging from operators to managers. Instead of creating a whole new methodology which would have to be introduced and taught it is easier to transfer a well established tool from its origin into a new area of application.
- *On-line availability.* As part of the *integrated simulation* concept LPC should be distributed to all persons in need without extra effort. It should be accessible and integrated within the standard on-line reporting system.
- *Focus on important measures.* Numerous input parameters and performance measures could possibly be controlled by quality control charts. It is the goal to limit the amount of charts to the *important* ones. Important in this context means either (i) strong influence on performance of the production line analyzed or (ii) power to detect detractors in performance.
- *Use available actual data.* The system should make use of already available data sources from the shop-floor-control system as far as possible.

¹⁰This acknowledges the possibility of quite different starting points for LPC.

There should be no need to maintain new systems for data collection. Sometimes it might be necessary, though, to enhance the current data collection¹¹, see section 11.5.2.

- *Use of available simulation model.* To gain consistency use should be made of the available model — currently the analytical simulation model described in chapter 3. This implies some assumptions like stationarity of the processes being analyzed.
- *Fast but conservative alarms.* Changes in line performance should lead to alarms quickly and directly. These should only be triggered, if the changes are statistically significant.
- *Effortless maintenance.* In today's manufacturing environments it is common to operate several logistical systems. Logistical process control charts are yet another decision support system. It should facilitate the production, but not increase the already high cost of maintenance. This directly implies that charts have to be created automatically without manual interference.

11.2.2. Control Charts for Process Time at Process Steps

The process time at process steps is an input parameter of the simulation model (see section 4.3.1 or table 6.1). As stated in the previous section a quality control chart for input parameters primarily has the function to control the validation of the simulation model, in other words, a quality control chart for input parameters is a statistical test for the null-hypothesis that the parameter for which the chart is created is correctly specified in the simulation model.

A quality control chart of the Shewhart type is a special form of a statistical test and thus it is exposed to two different kinds of errors: (i) rejecting the null-hypothesis though it is true (first type of error, alpha-error) and (ii) accepting the null-hypothesis though it is not true (second type of error, beta-error). A quality control chart constructs a decision rule, i.e. accepting that the observed parameter has not changed, if the sample mean stays within the pre-computed interval or reasoning that the parameter has

¹¹The analysis of maintenance related information like MTBF and MTTR has been precluded because at the time of writing the controlling system was being migrated and thus not available.

changed, if the mean falls out of the acceptable region. Concerning the two possible types of error, obviously, any choice among *good* decision rules should have something to do with the relative seriousness of the two types of errors (see [Win75, p.409]).

Quality control charts keep the alpha-error under control: A process is assumed to be under statistical control until one observes enough proof against this assumption. In case of quality control charts for input parameters this behavior is deliberately chosen for the following reasons:

- It is quite hard to keep the simulation model on an accepted level of validation. Though the model of *integrated simulation* proposes several means to facilitate the parameter collection, still some amount of manual work for updating parameters is needed. The alpha-error is equivalent to a *false alarm* whereas an alarm is equivalent to the statement that a parameter is not specified correctly. The focus should be clearly laid on correcting parameters which have a probability of $1 - \alpha$ to be specified falsely. Because of the large amount of input parameters it is already quite likely to observe some of these cases.
- Actual parameters are taken from the shop-floor-control system which is not specifically designed to serve as a data source for quality control charts. The estimation of parameters can be influenced by numerous types of possible measurement errors. In designing the acceptable region too small one gets higher probabilities of causing an alarm which is due to measurement errors like later or earlier in and out-claim of operators, respectively.
- The second type of error, i. e. deciding that the parameter is correctly specified (H_0), even though it is not (H_1), is serious and should be taken into account, though it is not taken as serious as the error of first type.

Measurement of Process Times

To be able to create quality control charts for process times at process steps the process time has to be measured using the data available from the shop-floor-control system. Figure 11.12 shows the times of a part which is claimed in and out at two consecutive process steps. Let in_{n-1} and in_n be the times of the claim-in at process steps n and $n - 1$, respectively and out_{n-1} and

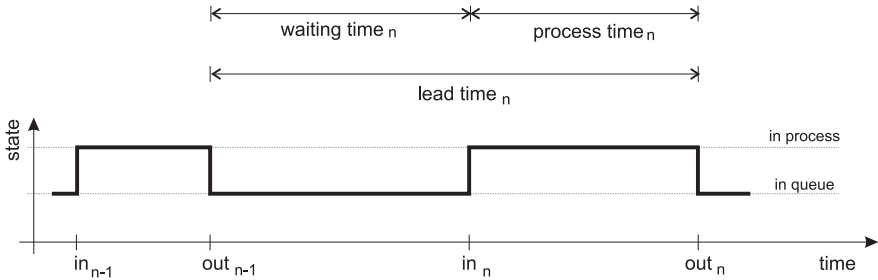


Figure 11.12.: In and out-claim at successive process steps.

out_n be the times of the out-claim accordingly. From these times the process time, waiting time, and lead time that a part observes at process step n can be derived by

- $process\ time_n = out_n - in_n$,
- $waiting\ time_n = in_n - out_{n-1}$,
- $lead\ time_n = out_n - out_{n-1}$.

The transaction table of the shop-floor-control system contains these times, but in a quite different format. Thus, the data has to be transformed before the process time observations can be calculated in the above fashion. As noted in section 11.2.7 the number of possible processes for which quality control charts should be generated is normally far less than the possible number of processes. The important processes therefore first have to be filtered out. This is all done by an SAS script which is also used to generate the control charts and publish them in HTML format. The following steps describe the procedures needed to calculate observations of process times:

1. To evaluate which processes should be included the mapping table stored in the EPOS database is read. This table contains the mapping between shop-floor-control machines and EPOS work centers. Moreover, the administrator can select for which machines a control chart is to be generated, see figure 11.16. Information for just these machines is retrieved.

2. Using the work center number the process steps for which charts are to be generated can be identified in the simulation model. The work center, operation, product group, and process step parameters are retrieved. These simulation parameters are later needed as plan parameters for the control charts.

To be able to filter the needed products another mapping table is needed to relate the product groups of the simulation model to the product types of the transaction table of the shop-floor-control system.

3. Now the information which transactions have to be selected from the transaction log are available. Using SAS macro variables where clauses for operation numbers, product type names, and machine identifiers are constructed. These clauses are then used in a query against the transaction log. Using a time frame which is normally set from three to nine months the needed transactions can be loaded.
4. A possible cause for autocorrelation in the time series of process times lies in batch processing. Two parts that are moved to and from a process step in the same transportation batch will almost inevitably have about the same claim times. Therefore process times of batches instead of individual parts are calculated. Due to the correlation between parts in a batch a sample of individual parts would increase the sample size but not the information contained in this sample.

For this reason two tables are created: the first table contains the last in-claim of a part for each batch at each process step. The second table contains the first out-claim accordingly. These points in time have been chosen because the time between the first and last in-claim at a machine is not assumed to be part of the process time but rather the preparation of the operator before the processing starts.

5. The two tables created during the last step are now joined in order to obtain one table of rows containing the time of the in and out-claim of a batch at a specific process step. From this the process time can directly be calculated as the difference between out and in-claim. Thus, the table contains all observation needed in the time frame for all specified process steps.

Concerning the measurement of process times, batch processing at machines creates a problem if the machine operates in sequential fashion. If

a machine operates in parallel the process time does not change with the number of parts processed¹². Sputter or ovens are examples of this type of machine. The process time does not change whether five or fifty parts are heated in the oven. For sequential machines the process time of a batch is highly dependent on the number of processed parts. Some machines take individual parts from an input buffer and process these part by part. Obviously the process time of the whole batch depends on the number of processed parts. To be able to compare planned and observed process times for sequential machines the process time has to be compared for a matching number of parts processed. While the simulation model contains the process time for a reference batch size the actual batch size cannot be determined from the transaction log which only contains information about the transport batch size which can differ from the batch size used during processing. Quality control charts for process times can therefore only be generated for parallel machines (see section 11.5.3 for a list of requirements for future shop-floor-control system implementations). Some important machines do operate in this fashion, nevertheless the missing batch size information strongly reduces the possible range of application.

Design of Quality Control Charts

After the observations of the process times have been calculated the control charts have to be generated. The design of a quality control chart involves decisions to use different types of charts, statistics used, or assumptions made. The following points describe the design decisions taken:

- The purpose of quality control charts for process times has been defined to test whether the process time which can be observed in the production is equal to the parameter specified in the simulation model. A modified Shewhart chart for sample means is chosen as the appropriate type of chart. Charts for medians or extreme values are not possible because the specification of the plan parameter only permits a test against the sample mean. Compared to other types of charts like CuSum or moving average charts Shewhart charts have the advantage of easy interpretation.

¹²Some parts of the processing time like set-up procedures might depend on the number of processed parts, but these effects are neglected for machines which are classified as parallel.

- The Shewhart charts are generated for a planned value μ_0 which is the parameter taken from the simulation model, i.e. the planned cycle time of the process step.
- For mean charts the type of subgroup for which a sample can be drawn has to be defined. It is chosen to use the week in which the process has been finished, i.e. the week in which the first part of a batch has been claimed out of operation. The shop-floor-control system contains information about all batches which have been processed at a specific process step. Instead of taking a sample from these observations all observations are used to calculate the sample mean. Calculating sample means from all observation of a week has the following reasons:
 1. This sample scheme permits to use all possible information without creating additional costs as the transactions are logged anyway.
 2. To be able to use a control chart of the Shewhart type the sample mean has to be normally distributed. This is the case if the source process is normally distributed which cannot be assumed for the process time at process steps (see analysis in section 11.1.5). For large sample sizes the sample mean is approximately normally distributed as an effect of the central limit theorem. This is the reason why even without normally distributed base processes Shewhart charts can successfully be deployed as long as the sample size stays large enough (see [Whe91] and [SAS99]). By using all possible observations the sample size can be kept high. Moreover, the choice of one week as subgroups ensures that enough parts have been passed each process step in order to create statistically valuable statements.
 3. One week seems to be an appropriate interval to check the validity of the simulation model.

Choosing a week as subgroup and using all observations to calculate the sample mean implies that no fixed sample size can be used to generate control charts. Instead for each week a possibly different acceptance interval has to be computed which becomes apparent with control limits which change over time.

- Traditional Shewhart charts assume the observations in the sample to be independent. Though it is not quite apparent, consecutive process times of batches do not need to be independent. If the machine's processing batch size is large compared to the transport batch size, it is likely for parts in consecutive transport batches to be processed in the same run of the machine (see section 11.1.5). The occurrence of correlation between the observations in the sample cannot be neglected (see [FK67]) as it can have a large effect on the length of the acceptable interval in the control charts. The handling of autocorrelation is therefore treated in the following section.

Generating Control Charts

The QC module of the SAS system contains a procedure which allows the creation of different types of control charts of the Shewhart type. The `XChart`-statement of the Shewhart procedure can be used to generate quality control charts for subgroup means. Using this statement the procedure is capable of computing various statistics including control limits and also of generating the graphical output of the desired charts. As the plan parameter μ_0 and the control limits are not derived in the standard way — due to the design decision stated in the previous section — the procedure is only used to draw the control charts. Prior to this the calculation of the chart content is done using standard SAS statistical methods.

Variable	Description
chart variable(s)	variable(s) that define the chart to which the parameters apply
subgroup variable	variable to distinguish between subgroups
._LCLX_	lower control limit for the mean
._UCLX_	upper control limit for the mean
._MEAN_	process mean μ_0
._SUBN_	subgroup sample size
._SUBX_	subgroup mean

Table 11.3.: Variables required in the SAS data set used to create Shewhart-type control charts

To use the **Shewhart**-procedure to create control charts from pre-calculated data the information has to be put into a special format — a SAS data set whose structure is shown in table 11.3. By creating and filling a SAS data set in the specified format it is possible to create control charts for various processes by just executing the **Shewhart**-procedure with the data set as an input parameter. One chart is created for each distinct combination found in the *chart* variables. For process time charts these variables are either

- work center, operation, and product group, or
- work center, and operation.

Often it is sufficient to distinguish just between the second combination, i. e. work center and operation. Including the product group is only reasonable if the planned process time of an operation at a work center differs between product groups. If planned process times of different product groups are equal, just one chart for the combination of work center and operation should be created. This decreases the number of control charts while increasing the sample sizes in the combined chart.

For each chart to be created¹³ there are observations $x_{i,j}$ with i as the index of the subgroup and j as the index within a subgroup. Thus, observations

$$\begin{array}{cccc} x_{1,1}, & x_{1,2}, & \dots, & x_{1,n_1} \\ x_{2,1}, & x_{2,2}, & \dots, & x_{2,n_2} \\ \vdots & & & \ddots \\ x_{m,1}, & x_{m,2}, & \dots, & x_{m,n_m} \end{array} \quad (11.31)$$

are available whereas the sizes of the subgroups n_i for i, \dots, m do not need to be equal. The size of the whole sample is $n = \sum_{i=1}^m n_i$.

The first step to fill the SAS data set needed is to create one record for each point that is to be shown in the chart and fill in the calculated subgroup means (`_MEANS_`) and subgroup sample sizes (`_SUBN_`). As an estimator for the subgroup mean

$$\bar{x}_j = n_j^{-1} \sum_{i=1}^{n_j} x_{i,j} \quad \text{for } j = 1, \dots, m \quad (11.32)$$

¹³Without loss of generality and to facilitate notation the creation of one specific chart is presented, the implementation uses an outer loop to apply every step presented to all charts to be created.

is used. It is assumed that the observed process is in stationary state — an assumption which is supported by the analysis performed in section 11.1.5. Moreover, if the production line is not restarted or just opened for production, there is no initial phase which would normally appear in the analysis of simulation-generated output. This initial phase would bias the estimator (see [LK84]) which remains unbiased in this case.

Instead of using the planned process time as target μ_0 (`_MEAN_`) the mean completion time of the simulation model is chosen. Though the process time is to be controlled the estimates from the shop-floor-control system are actually completion times because machine outages and other types of down times are included between the in and out-claim. The corresponding parameter on the plan side can therefore be identified as the mean completion time which is calculated by the model (see section 3.10). Thus, the data collected by the shop-floor-control system does not allow to monitor the process time directly.

In previous steps the mapping table between the shop-floor-control system and EPOS has been used to identify the completion time for each chart. At this step it is therefore just a matter of setting this value for every subgroup.

Calculating the Control Limits

A control chart is a statistical test based on the sample mean for every subgroup in the chart. It is tested whether the null-hypothesis stating that the average process time observed equals the planned value has to be accepted or if enough evidence exists to reject this hypothesis, formally written as

$$H_0 : \mu_0 = \mu_i \quad (11.33)$$

$$H_1 : \mu_0 \neq \mu_i \quad (11.34)$$

for $i = 1, \dots, m$. For Shewhart type charts the planned value is drawn as a horizontal line around which control limits are specified. These form the acceptable region. If the sample mean \bar{x} lies within this region, the null-hypothesis H_0 is accepted, otherwise the alternative H_1 is chosen.

The probability that \bar{x} lies within a region around μ_0 can be expressed by

$$P\left(\mu_0 - \lambda \frac{\sigma}{\sqrt{n_i}} < \bar{x}_i < \mu_0 + \lambda \frac{\sigma}{\sqrt{n_i}}\right) = 1 - \alpha. \quad (11.35)$$

Assuming the process is normally distributed, λ which defines the size of the region around μ_0 can be derived for a specific level of significance α by

$$P\left(\mu_0 - \lambda \frac{\sigma}{\sqrt{n_i}} < \bar{x}_i < \mu_0 + \lambda \frac{\sigma}{\sqrt{n_i}}\right) = P\left(-\lambda < \frac{\bar{x}_i - \mu_0}{\sigma} \sqrt{n_i} < \lambda\right) \quad (11.36)$$

$$= \Phi(\lambda) - \Phi(-\lambda) \quad (11.37)$$

$$= 2\Phi(\lambda) - 1. \quad (11.38)$$

This is combined with (11.35) to obtain $\lambda = \Phi^{-1}(1 - \alpha/2) = z_{1-\alpha/2}$. Using tables of the standard normal distribution λ is determined to be 2.58 for a traditionally used level of significance $\alpha = 0.01$. The upper and lower control limits, c_u, c_l , respectively, can thus be set as

$$c_u = \mu_0 + z_{1-\alpha/2} \frac{\sigma}{\sqrt{n_i}} \quad (11.39)$$

$$c_l = \mu_0 - z_{1-\alpha/2} \frac{\sigma}{\sqrt{n_i}} \quad (11.40)$$

So far, this derivation has shown the standard background for quality control charts of the Shewhart type. The logistical processes like the process time or the waiting time (which is discussed in the following section) violate two important assumptions that the model is based on. As it is shown in section 11.1.5 strong statistical evidence exists that logistical processes are neither normally distributed, nor are the observations in the sample independent. The former problem is not as severe as the latter, because — as an effect of the central limit theorem — the sample mean is with large sample sizes approximately normally distributed¹⁴.

The existence of autocorrelation creates a problem when deriving (11.36) from (11.38). For this step a standardized¹⁵ normally distributed random variable is needed. This is created in step (11.36) as for each random variable X the term $(X - E[X])/\sqrt{\text{Var}[X]}$ denotes a standardized random variable. If the observations in the sample are correlated, the variance of the sample mean $\text{Var}[\bar{X}]$ is not equal to σ^2/n as it would be for independent observations. It can be approximated by the following formula which has been

¹⁴By analyzing the structure of the autocorrelation in the process Diananda [Dia53] shows that the central limit theorem holds for the sample mean.

¹⁵A random variable Y is standardized if $E[Y] = 0$ and $\text{Var}[Y] = 1$.

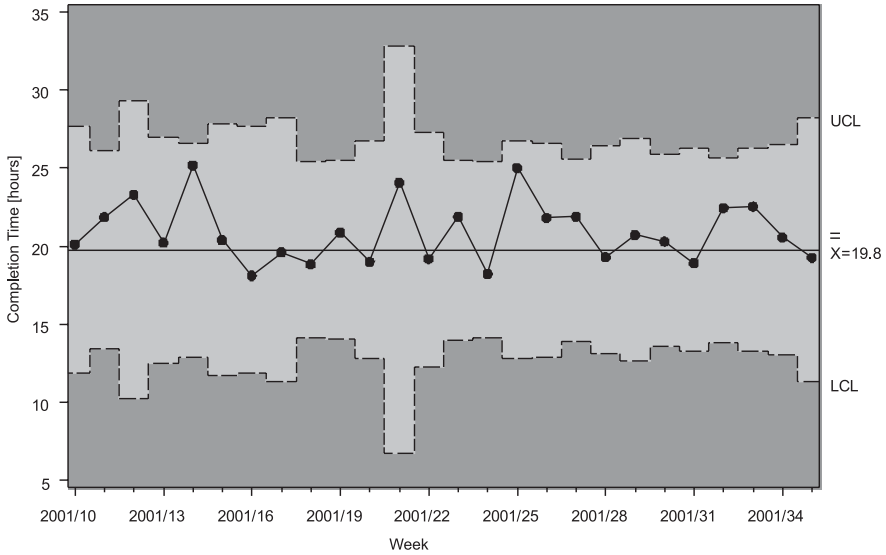


Figure 11.13.: Quality control chart of the process time of the overcoat sputter process performed on a typical parallel sputter machine.

introduced in section 11.1.5:

$$\text{Var}[\bar{X}] = \frac{C}{n} \approx \frac{\sigma^2}{n} \sum_{k=-\infty}^{\infty} \rho(k). \quad (11.41)$$

The factor $\sum_{k=-\infty}^{\infty} \rho(k)$ can be identified as the effect that correlation between the observations in the sample has on the reliability of the sample mean. In other words, this factor is the number of correlated observations equivalent to one independent observation and $N / (\sum_{k=-\infty}^{\infty} \rho(k))$ is known as the equivalent independent sample size. (see [Lav83b, p. 305]).

Various techniques have been proposed to estimate the variance of the sample mean by coping with the amount of autocorrelation or by arranging the observations in order to obtain independence among them, see section 11.2.3. Neither of the presented techniques is applicable to such small sample sizes as they occur within each subgroup of the control chart because

the sample size is small compared to the covariance structure of the process. Methods trying to arrange observations to obtain independent random variables cannot succeed because the correlation between even the first and the last observation in the sample is likely to be high. Methods trying to analyze the correlation directly cannot succeed because there are not enough observations to estimate the autocorrelation. It is desirable, though, to have short intervals between calculations of the sample mean as the main purpose of the chart is the fast detection of changes in the observed process. The size of subgroups should therefore not be increased in order to gain larger samples.

This problem is attacked by the following approach: An estimation method coping with autocorrelation is applied on the whole information available for a control chart yielding an estimate of the variance of the sample mean $\text{Var}[\bar{X}]$. It can be shown that for processes in steady state this variance decreases linearly with the run length, formally $\text{Var}[\bar{X}] = C/n$ (see equation (11.41), [GH74, p. 419], and [FK67]) Thus, the estimation method is used to obtain C . Assuming the variance of the underlying process does not change between subgroups this estimate can be employed to calculate control limits for varying sample sizes. If the variance of the sample mean $\text{Var}[\bar{X}]$ has been estimated by some technique, the upper and lower control limits c_u and c_l for each subgroup, respectively, can be obtained by

$$c_u(k) = \mu_0 + z_{1-\alpha/2} \sqrt{\frac{\text{Var}[\bar{X}]k}{n}} \quad (11.42)$$

$$c_l(k) = \mu_0 - z_{1-\alpha/2} \sqrt{\frac{\text{Var}[\bar{X}]k}{n}} \quad (11.43)$$

whereas k denotes the size of the subgroup in the control chart. These are set to the variables `_UCLX_` and `_LCLX_` of the SAS data set, respectively.

This completes the data set and it can be used by the **Shewhart**-procedure to create all control charts requested. Via SAS ODS¹⁶, HTML files with corresponding GIF-graphics easily be created and uploaded onto an HTTP-server which publishes the control charts in the intranet.

¹⁶Output Delivery System

11.2.3. Methods for Estimating the Variance of the Sample Mean

The analysis of both — process time and the waiting time — processes showed that they contain a substantial amount of autocorrelation which cannot be neglected when estimating the sample mean. In the literature the following techniques to tackle the problem of autocorrelation have been proposed

- *Independent replications.* Simulating a model numerous times using different random number streams in order to gain independent observations.
- *Batch means.* Forming batches which are approximately independent because of the batch size which is large compared to the covariance structure of the process.
- *Autoregressive representation.* Fitting the observed process to an autoregressive model and estimating the variance of the sample mean from the parameters of the fitted model.
- *Spectral analysis.* Estimating the spectral density which contains the same information as the covariance function.
- *Regeneration cycles.* Gaining independent observations by splitting the simulation run into several cycles separated by regeneration points, i. e. points at which the process restarts with the same statistical properties.

The method of independent replications is only applicable, if it is possible to replicate the stochastic process. While replicating the simulation of a model can easily be achieved, the production process can obviously not be replicated. Two of the remaining methods, namely batch means and regeneration cycles, group the observation into batches or cycles which are (approximately) independent. The methods of spectrum analysis and autoregressive representation directly cope with autocorrelation in the estimation process.

Before discussing the advantages and disadvantages of these methods the important ones are briefly introduced in the following. For further information the reader is directed to Law and Kelton [LK84, LK91].

Batch Means

The method of batch means employs standard statistical techniques designed for independent samples. To obtain an approximately independent sample from an autocorrelated time series the observations are arranged in a fashion that diminishes the effect of autocorrelation. The time series x_1, \dots, x_n is partitioned into M sequences of equal length L , formally $x_{(m-1)L+1}, \dots, x_{mL}$ for $m = 1, \dots, M$. Let

$$\hat{x}_m = \frac{1}{L} \sum_{i=(m-1)L+1}^{mL} x_i \quad \text{for } m = 1, \dots, M \quad (11.44)$$

be the sample mean of the m th batch. If the length of the sequences L is chosen large enough regarding the covariance structure of the process, the M batch means \hat{x}_m are approximately independent realizations of random variables (see [Lav83b, p. 307] for further information on the size of L). It can easily be shown that the estimator

$$\hat{x} = \frac{1}{M} \sum_{i=1}^M \hat{x}_i \quad (11.45)$$

is equal to the mean of the time series x_1, \dots, x_n and thus is an unbiased estimator for the expectation of the underlying process. Its variance can be estimated by

$$\text{Var}[\hat{x}] = \frac{1}{M-1} \sum_{i=1}^M (\hat{x}_i - \hat{x})^2. \quad (11.46)$$

This estimator can be used to generate confidence intervals or the control limits of control charts.

Spectral Analysis

The methods for estimating confidence intervals based on spectral analysis differ from the already presented methods. Instead of trying to use classical statistical procedures on independent observations these methods directly generate estimates and correct these from the effect of autocorrelation.

The basic idea of spectral methods lies in the relation between the autocovariance function $\gamma(k)$ for lag $k = 0, 1, \dots, \infty$ and the power spectral

density of autocovariance stationary processes. For these processes with $\sum_{k=-\infty}^{\infty} |\gamma(k)| < \infty$ it can be shown that

$$\gamma(k) = \int_{-\pi}^{\pi} f(\lambda) e^{i\lambda k} d\lambda \quad (11.47)$$

holds¹⁷. The function $f(\lambda)$ describes the contribution that a frequency λ makes to the overall variance of the process (X_t) ¹⁸ and the variance of the process σ^2 can therefore be written as

$$\sigma^2 = \gamma(0) = \int_{-\pi}^{\pi} f(\lambda) d\lambda. \quad (11.48)$$

The nonnegative function $f(\lambda)$ is called power spectral density and defines the autocovariances $\gamma(k)$. Inversely, $f(\lambda)$ is defined by the autocovariance through

$$f(\lambda) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma(k) e^{-i\lambda k} \quad \text{for } -\pi \leq \lambda \leq \pi \quad (11.49)$$

$$= \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma(k) (\cos(-\lambda k) + i \sin(-\lambda k)). \quad (11.50)$$

One is interested in the real part of this which computes to

$$f(\lambda) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma(k) \cos(\lambda k) \quad (11.51)$$

$$= \frac{\gamma(0)}{2\pi} + \frac{1}{\pi} \sum_{k=1}^{\infty} \gamma(k) \cos(\lambda k) \quad (11.52)$$

using the symmetry of the covariance, $\gamma(k) = \gamma(-k)$. Evaluating the power spectral density at frequency 0 yields

$$f(0) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma(k). \quad (11.53)$$

¹⁷With i as the imaginary unit, defined by $i^2 = -1$

¹⁸As usual, let X_t denote the random variables that form a time series and x_t be a specific realization of such a process.

When this result is set in relation to (11.30) one finally gets

$$\text{Var}[\bar{X}] \approx \frac{1}{n} \sum_{k=-\infty}^{\infty} \gamma(k) = \frac{f(0)}{T} \quad (11.54)$$

where T denotes the length of the sample interval in radians, i. e. $T = n/2\pi$. Estimating the variance of the sample mean is therefore related to estimating the power spectral density at zero frequency.

If the spectral density is to be estimated from a time series x_1, \dots, x_n , the theoretical covariances $\gamma(k)$ can be replaced by their estimates $\frac{n}{n-k}c(k)$. This leads to the sample spectrum

$$\hat{f}(\lambda) = \frac{c(0)}{2\pi} + \frac{1}{\pi} \sum_{k=1}^{n-1} c(k) \cos(\lambda k). \quad (11.55)$$

However, for lag k near n the covariance estimators would be based on only a few observations and hence have poor statistical properties. Formally, the estimator $\hat{f}(\lambda)$ is asymptotically unbiased, but not consistent, i. e. its variance does not decrease with the sample size n (see [Har93, p. 711]).

To obtain a consistent estimator one averages the sample spectrum by using so called spectral windows. Each estimate $\gamma(k)$ is weighted with a weighting function $w(k)$ which leads to

$$\hat{f}_M(\lambda) = \frac{c(0)}{2\pi} + \frac{1}{\pi} \sum_{k=1}^M w_M(k) c(k) \cos(\lambda k). \quad (11.56)$$

Note that this estimator contains M instead of $n - 1$ lags. If n is sufficiently large, then there exists a quantity M such that for all $k > M$, $\gamma(k) \approx 0$. The summation will therefore not appreciably change, if more than M lags are included. Moreover, to gain a consistent estimator, $\lim_{n \rightarrow \infty} M/n = 0$ has to be satisfied, see Fishman [Fis67].

Different weighting functions have been proposed, the most important ones are the Bartlett window defined as

$$x_M(k) = \begin{cases} 1 - \frac{k}{M} & \text{for } k = 0, \dots, M \\ 0 & \text{for } k > M, \end{cases} \quad (11.57)$$

the Tukey window with

$$x_M(k) = \begin{cases} \frac{1}{2} + \frac{1}{2} \cos\left(\frac{\pi k}{M}\right) & \text{for } k = 0, \dots, M \\ 0 & \text{for } k > M, \end{cases} \quad (11.58)$$

and finally the Parzen window defined as

$$x_M(k) = \begin{cases} 1 - 6 \left(\frac{k}{m}\right)^2 + 6 \left(\frac{k}{m}\right)^3 & \text{for } 0 \leq k < m/2 \\ 2 \left(1 - \frac{k}{m}\right)^3 & \text{for } m/2 \leq k \leq M \\ 0 & \text{for } k > M. \end{cases} \quad (11.59)$$

Using one of these windows and (11.56) it becomes possible to estimate the power spectral density. The parameters of this method are the choice of window and the length of summation M . This is the classical estimation method, compare [Fis67, Fis73, Har93, LK84].

Estimation Method proposed by Heidelberger and Welch

Heidelberger and Welch [HW81] note that the power spectrum — a function which is symmetric around zero — either has a peak or a valley at zero frequency. It is therefore not approximately linear and for many of the processes normally found in queueing systems the spectrum is sharply peaked at zero frequency. Hence any weighted average of the spectrum will result in a biased estimate of $f(0)$. If the width of the spectral window is decreased to minimize the bias, the variance of the estimator is increased proportionally with the decrease of the width. One can therefore either obtain a biased estimate that is stable or an unbiased estimate that is highly variable. The estimation technique presented by Heidelberger and Welch tries to circumvent this dilemma with a different approach which is presented in the following.

The estimation method uses the periodogram $I(\lambda)$ of the time series X_1, \dots, X_n as the starting point. It is defined by

$$I_n(i) = \frac{1}{n} \left| \sum_{j=1}^n X_j e^{-2\pi i(j-1)/n} \right|^2 = \frac{1}{n} |a_i|^2 \quad \text{for } i = 1, \dots, n/2 \quad (11.60)$$

where a_i denotes the discrete Fourier transform of the time series (for more information about the discrete Fourier transformation the interested reader is directed to [PTVF88], for example). The periodogram is an unbiased estimator of the power spectrum density, i. e.

$$E[I_n(i)] = f\left(\frac{i}{n}\right) \quad \text{for } i = 1, \dots, n/2. \quad (11.61)$$

It has further interesting properties, e.g. while the time series X_1, \dots, X_n maybe highly correlated, the periodogram is approximately uncorrelated.

Figure 11.14 shows an example of a periodogram. In (a) the time series of the waiting time process in an $M/M/1$ system with $\lambda = 0.8$, $\mu = 1.0$ is shown. The corresponding periodogram is depicted in (b). Assuming that the spectral density is a smooth function around frequency zero, regression techniques can be employed. The spectral estimate then is the y-axis intercept of the fitted regression function. This approach circumvents the bias problem of the spectral window method which only yields an unbiased estimate if the region around zero is flat. Two problems still remain:

1. The variance of the periodogram is not constant over the entire domain, but rather $\text{Var}[I(\lambda)] \approx f^2(\lambda)$. This effect can be eliminated by using the logarithm of the periodogram as basis for the regression. The variance become $\text{Var}[I(\lambda)] \approx 1.645$. Moreover, $\log(I(\lambda))$ is a smoother function than $I(\lambda)$.
2. The distribution of the periodogram is very positively skewed, after taking the logarithm the distribution becomes very negatively skewed, compare figure 11.14.b and 11.14.c. By averaging over adjacent values of the periodogram before taking the logarithm, this skewness can be decreased, see figure 11.14.d. Moreover, this also decreases the variances.

Heidelberger and Welch studied different polynomial regressions with parameters K as the number of points used in the polynomial fit and d as the degree of the polynomial. The parameter K determines the frequency range over which the fit is made, it is equal to $[0, 2K/n]$. The author empirically investigated the choice of parameters and recommend $K = 25$ for small and $K = 50$ for large sample sizes. Using linear regression ($d = 1$) proved to be inadequate for practical purposes.

Regeneration Cycles

The method of regeneration cycles just like the method of batch means generates independent or approximately independent sequences of the underlying time series by means of an experimental protocol. Whereas the method of

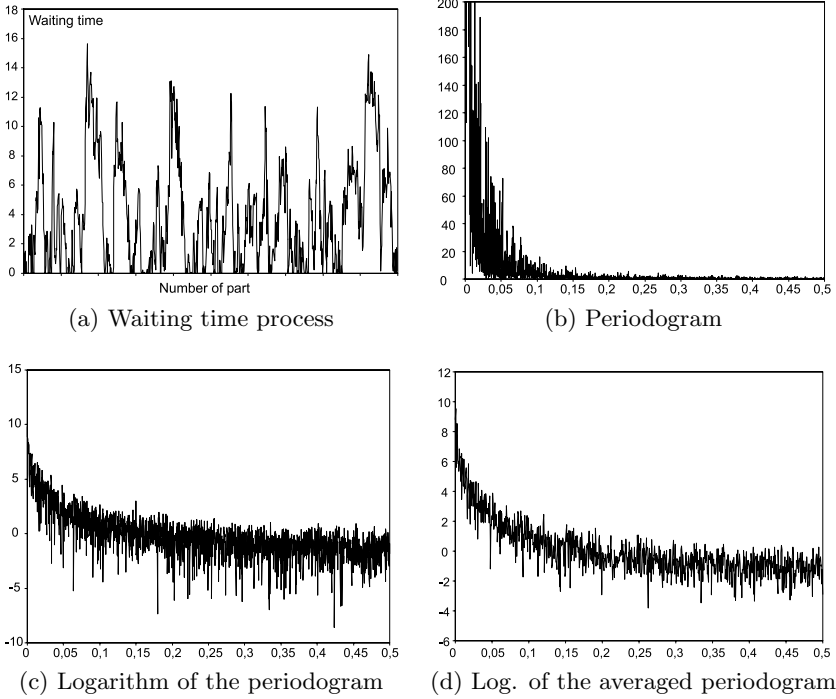


Figure 11.14.: Periodograms

batch means produces approximately independent sequences of equal length, the method of regeneration cycles provides independent sequences of different lengths. Regeneration cycles have the strongest theoretical foundations of all methods presented that work with single run output. It cannot be applied to all time series as it requires a certain stochastic structure which does not exist for all models.

With the process $(X_t)_{t \in \mathbb{N}}$ let $1 \leq R_1 < R_2 < \dots$ be a set of random time points. With these points it is possible to arrange the process (X_t) into sequences $X_{R_i}, X_{R_{i+1}}, \dots, X_{R_{i+1}-1}$ of length $L_i = R_{i+1} - R_i$ for $i \in \mathbb{N}$. Sometimes it is possible to show that a stochastic process (X_t) regenerates at such points R_i , i.e. the process starts afresh and the distribution of the processes $(X_{R_i+j})_{j \in \mathbb{N}_0}$ are the same for $j \in \mathbb{N}$. The points R_i are then called

regeneration points, the subsequences, i.e. the portions of the process between the regeneration points, are denoted as regeneration cycles or tours. The essential idea of the regenerative method is that between any two successive regeneration points the evolution of the process is a probabilistic replica of the process between any other such pair of regeneration points (compare [IS80], [Kin72], and [Coh76]).

The length L_i of each tour is a random variable, just as the sum over each tour defined by

$$S_j = \sum_{i=R_j}^{R_{j+1}-1} X_i \quad \text{for } j \in \mathbb{N}. \quad (11.62)$$

Under mild regularity conditions the random vectors $U_j = (S_j, L_j)$ are independent and identically distributed which follows directly from the definition of a regenerative process, see [Cin75, p. 298] and [IS80, p. 15]. It can be shown that under very general conditions the steady state mean $E[X]$ of the regenerative process $(X_t)_{t \in \mathbb{N}}$ is

$$E[X] = \frac{E[S_j]}{E[L_j]} \quad \text{for } j \in \mathbb{N}. \quad (11.63)$$

Let x_1, \dots, x_n be a time series generated by the regenerative process (X_t) consisting of k tours which are separated by regeneration points $1 \leq r_1 < r_2 < \dots < r_{k+1}$ with $r_i \in \{1, \dots, n\}$ for $i = 1, \dots, k+1$. The problem of estimating $\mu = E[X]$ by the regenerative method is then equal to the problem of estimating the ratio $E[S_j]/E[L_j]$ from k independent, identically distributed pairs of observations $(S_1, L_1), \dots, (S_k, L_k)$ whereas S_j and L_j are in general not independent, i.e. $\text{Cov}[S_j, L_j] \neq 0$ for $j = 1, \dots, k$. A biased, but asymptotically unbiased estimator of the ratio is given by

$$\hat{\mu} = \frac{\frac{1}{k} \sum_{j=1}^k S_j}{\frac{1}{k} \sum_{j=1}^k L_j} = \frac{\hat{S}}{\hat{L}}, \quad (11.64)$$

see [IS80, theorem 2.4.1] and [Lav83b, p. 314]. To construct a confidence interval a sequence of random variables Z_j is defined by

$$Z_j = S_j - \mu L_j \quad \text{for } j = 1, \dots, k. \quad (11.65)$$

The random variables Z_j are independent, identically distributed with zero mean

$$\mathbb{E}[Z_j] = \mathbb{E}[S_j] - \mu \mathbb{E}[L_j] = 0 \quad (11.66)$$

and variance

$$\sigma_Z^2 = \text{Var}[S_j] - 2\mu \text{Cov}[S_j, L_j] + \mu^2 \text{Var}[L_j] \quad \text{for } j = 1, \dots, k. \quad (11.67)$$

With

$$\hat{Z} = \frac{1}{k} \sum_{j=1}^k Z_j = \frac{1}{k} \sum_{j=1}^k S_j - \mu \frac{1}{k} \sum_{j=1}^k L_j = \hat{S} - \mu \hat{L}, \quad (11.68)$$

increasing number of tours k , and as a result of the central limit theorem the distribution of

$$\frac{\hat{Z} - \mathbb{E}[Z_j]}{\sigma_Z/\sqrt{k}} = \frac{\hat{Z}}{\sigma_Z/\sqrt{k}} \quad (11.69)$$

converges to the standard normal distribution. The right hand side of (11.69) can be written as

$$\frac{\hat{Z}}{\sigma_Z/\sqrt{k}} = \frac{\hat{S} - \mu \hat{L}}{\sigma_Z/\sqrt{k}} = \frac{\hat{\mu} - \mu}{\sigma_Z/\sqrt{k}}. \quad (11.70)$$

This result can be used to construct the confidence interval for the estimator $\hat{\mu}$. The variance σ_Z^2 is not known, but can be estimated by using the standard estimators on (11.67). Even by estimation of σ_Z

$$\frac{\hat{\mu} - \mu}{\sigma_Z/\sqrt{k}} \quad (11.71)$$

is approximately standard normally distributed for large number of tours k and the approximate confidence interval statement is formed by

$$\mathbb{P} \left\{ \hat{\mu} - \Phi^{-1}(1 - \alpha/2) \frac{\sigma_Z}{\hat{L}\sqrt{k}} \leq \mu \leq \hat{\mu} + \Phi^{-1}(1 - \alpha/2) \frac{\sigma_Z}{\hat{L}\sqrt{k}} \right\} \approx 1 - \alpha. \quad (11.72)$$

11.2.4. Experimental Results

To analyze the performance of the relevant methods experimental studies of different stochastic processes have been conducted. The performance of each

method is measured by the observed coverage of the estimated confidence intervals. For each process a measure for which the expectation could be analytically derived has been chosen as the target. The process has then been simulated for $R = 500$ replications and the percentage of the target lying in the 90% confidence interval, the relative half length of the confidence interval, and the variance of the interval's half length have been recorded.

Stochastic Processes

The first process studied is white noise, i.e. the process $(X_t)_{t \in \mathbb{N}}$ has the following properties

$$E[X_t] = 0, \quad \text{Var}[X_t] = \sigma^2 \quad \text{for all } t \in \mathbb{N}, \quad (11.73)$$

$$E[X_{t_1}X_{t_2}] = 0 \quad \text{for all } t_1 \neq t_2. \quad (11.74)$$

The process (X_t) therefore has expectation 0, its variance is constant, and all random variables X_t are uncorrelated. In the experiment the random variables X_t are uniformly distributed in $[0, 100]$, i.e. $X_t \stackrel{d}{=} \mathcal{U}(0, 100)$, the target is chosen as $E[X_t] = 50$.

While white noise can be characterized by non-existing autocorrelation, the second process, a first order autoregressive process, is chosen specifically to contain autocorrelation. Formally, a stochastic process (Y_t) with

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p} + \epsilon_t \quad \text{for all } t \quad (11.75)$$

where $\phi_j \neq 0$ for $j = 1, \dots, p$ are parameters and ϵ_t is white noise is called autoregressive process of order p , or $\text{AR}(p)$ process. The second process in the experiment is an $\text{AR}(1)$ process with $\phi_1 = 0.9$ and $\epsilon_t \stackrel{d}{=} \mathcal{U}(-1, 1)$ for all t . The target is the expectation $E[X_t] = 0$.

To test the performance of the estimation methods in a problem-related field a process that occurs in queuing systems is included in the experiment. Let $(Q_t)_{t \in \mathbb{N}}$ be the queuing time process, i.e. Q_t denotes the queuing time of the t th part. For a queuing model $M/M/1$ with arrival rate λ , service rate μ , the steady state expectation of the queuing time can be analytically derived by

$$E[Q] = \lim_{t \rightarrow \infty} E[Q_t] = \frac{\varrho^2}{\lambda(1 - \varrho)}. \quad (11.76)$$

The parameters of the simulated queueing system are $\lambda = 0.8$ and $\mu = 1.0$, thus $\rho = 0.8$. The expectation $E[Q] = 4$ is set as the target for which the confidence intervals are to be estimated.

Let $x_1, \dots, x_N, y_1, \dots, y_N$, and q_1, \dots, q_N be time series of the stochastic processes $(X_t)_{t \in \mathbb{N}}$, $(Y_t)_{t \in \mathbb{N}}$, and $(Q_t)_{t \in \mathbb{N}}$, respectively. For different lengths N these time series are analyzed by

- the classical statistical method of estimating the variance of the sample mean, $\text{Var}[\bar{X}] = \sigma^2/N$, thus ignoring autocorrelation in the observed process,
- the method of batch means for different batch sizes b ,
- the classical spectral method of estimating $p(0)$ by summing over empiric autocorrelations for different upper limits of the summation index,
- the spectral analysis method proposed by Heidelberger and Welch using a fast Fourier transformation and quadratic regression model for $K = 50$ points, and
- the method of regeneration cycles. This estimation scheme could only be deployed for the last process, the waiting time in an $M/M/1$ system. The system is stochastically restarted at points when no parts are in the system.

Results

Table 11.4 shows the results of the three studied processes for sample sizes N ranging from 500 to 5000¹⁹. For the method of batch means and spectral analysis different batch sizes and run lengths, respectively, for each sample size are deployed. For this the sample size N is partitioned into k parts, $k = 10, 6, 4$, leaving the batch size b as $b = N/k$.

For each experiment three figures are shown. In the first row the coverage, i.e. the proportion of the overall replications ($R = 500$) in which the estimated confidence interval covered the analytically determined target, is shown. The second row shows the average relative half width of the confidence interval with the exception of the autoregressive model whose target

¹⁹The study has later been extended for sample sizes up to 15000 observations. The results have been similar for the extended studies.

N	σ^2/n	Batch means			Spectral analysis			Heid./Welch	Reg. cycles
		k=10	k=6	k=4	k=10	k=6	k=4		
White noise									
500	0.885	0.9	0.93	0.93	0.875	0.925	0.92	0.95	
1000	0.0426	0.0465	0.0508	0.0565	0.0427	0.0433	0.0437	0.0643	
	0.0019	0.3139	0.6567	1.2924	0.1528	0.2894	0.3364	0.4897	
	0.895	0.885	0.885	0.865	0.865	0.87	0.88	1.0	
	0.2998	0.0327	0.0363	0.0397	0.0301	0.0312	0.0306	0.0308	
1500	0.0005	0.1627	0.3641	0.7483	0.0801	0.1430	0.2079	0.3089	
	0.91	0.9	0.9	0.905	0.885	0.88	0.87	1.0	
3000	0.0266	0.0266	0.0276	0.0345	0.0247	0.0243	0.0252	0.1697	
	0.0001	0.0877	0.2374	0.3989	0.0488	0.0834	0.1238	1.4103	
5000	0.88	0.885	0.925	0.885	0.845	0.925	0.88	1.0	
	0.0173	0.0188	0.0211	0.0239	0.0173	0.0181	0.0176	0.1729	
	0.0001	0.0476	0.1153	0.2351	0.0216	0.0476	0.0577	0.9589	
	0.94	0.9	0.925	0.9	0.9	0.89	0.88	1.0	
	0.0134	0.0147	0.0163	0.0187	0.0135	0.0139	0.0144	0.2491	
	0.0252	0.0252	0.0618	0.1513	0.0138	0.0261	0.0361	1.1216	
AR(1) process									
500	0.28	0.85	0.91	0.88	0.84	0.875	0.855	0.905	
1000	1.3716	1.3716	1.8449	1.7133	2.4958	1.7483	1.418	6.6514	
	0.0002	0.0076	0.0223	0.0252	0.0045	0.0111	0.0161	0.0324	
	0.3	0.85	0.885	0.91	0.85	0.86	0.88	0.88	
1500	1.8788	1.0325	1.3252	1.1424	1.4032	1.3265	0.9723	0.9321	
	2.1e-5	0.0052	0.0110	0.0314	0.0030	0.0056	0.0077	0.0062	
	0.25	0.85	0.905	0.89	0.835	0.86	0.845	0.82	
3000	0.0207	1.8815	1.7942	1.1325	1.5527	1.4153	1.3263	1.8831	
	0.1e-6	0.0037	0.0083	0.0168	0.0018	0.0035	0.0050	0.0040	
	0.27	0.89	0.87	0.93	0.895	0.85	0.89	0.85	
5000	0.2298	0.8779	3.9309	0.4523	0.0877	1.7906	0.5211	1.0745	
	2.1e-6	0.0020	0.0041	0.0093	9.8e-4	0.0013	0.0025	0.0017	
	0.33	0.92	0.9	0.915	0.885	0.865	0.92	0.82	
	0.1990	1.3911	1.2409	1.3252	1.0958	0.7203	0.9245	1.3112	
	7.9e-7	9.9e-4	0.0023	0.0060	5.8e-4	0.0011	0.0016	0.0010	
Waiting time									
500	0.17	0.73	0.645	0.715	0.705	0.59	0.68	0.765	0.63
1000	0.0801	0.4287	0.5116	0.5887	0.3902	0.4281	0.4420	0.4853	0.4921
	0.0094	0.7882	3.6295	3.1061	0.6068	2.3291	1.4196	0.6134	0.9829
	0.08	0.71	0.745	0.75	0.66	0.695	0.69	0.78	0.71
1500	0.5853	0.3744	0.4467	0.4986	0.3408	0.3760	0.3756	0.4529	0.3881
	0.0055	0.0066	1.9165	3.7961	0.7857	1.2459	1.5674	1.0758	0.7469
	0.125	0.805	0.81	0.835	0.775	0.775	0.795	0.855	0.82
3000	0.0491	0.3575	0.3954	0.4651	0.3284	0.3364	0.3515	0.3954	0.3736
	0.0036	0.7659	1.4501	1.3658	0.6095	0.9611	0.7097	1.3476	0.7177
	0.105	0.795	0.795	0.815	0.78	0.78	0.8	0.86	0.78
5000	0.0357	0.2932	0.3234	0.3686	0.2717	0.2773	0.2825	0.3713	0.2667
	0.0016	0.6147	0.6045	1.3816	0.6147	0.3951	0.6677	0.9996	0.2383
	0.155	0.83	0.865	0.88	0.835	0.83	0.86	0.915	0.86
	0.0278	0.2314	0.2638	0.3121	0.2159	0.2767	0.2380	0.3195	0.2319
	4.8e-4	0.2383	0.2767	0.6497	0.1935	0.1691	0.2892	0.4397	0.0970

Table 11.4.: Experimental results of five different methods for confidence interval generation applied to three different stochastic processes. See section 11.2.4 on page 407 for further explanation of this table.

is zero. There the average half width of the confidence interval is stated. The third row shows the variance of the width of the estimated confidence intervals.

For deployment of an estimation technique the most important measure of performance seems to be the coverage: If the 90% confidence interval is calculated, then in 90% of all cases the true value should lie within the interval. Any discrepancy from the desired level of confidence $1 - \alpha$ is either misleading or hinders decision-making. The confidence interval is misleading, if the true coverage is less than the specified level. A coverage that lies above the specified level is caused by half lengths which are too large and obviously lower the value of the information provided.

For the white noise process all estimation methods with the exception of the method proposed by Heidelberger/Welch perform well. The method by Heidelberger and Welch creates confidence intervals whose half widths are too large which becomes extremely apparent at large sample sizes. The effects are coverage figures around 100%.

The classical σ^2/N method neglecting the existence of autocorrelation cannot be successfully employed for autocorrelated time series. This can be seen in the results of the autoregressive and the queueing model. For the former model the method returns confidence intervals around 30% coverage and around 10% for the latter model.

The method of batch means and spectral analysis perform equally for the autoregressive model. In the case of the queueing model the method of batch means seems to be slightly advantageous. The conclusion by other authors that both methods perform better for larger batch sizes can be observed in most cases. Moreover, both methods seem to perform better for larger batch sizes.

The technique by Heidelberger/Welch returns close coverage results for all autocorrelated time series. For the queueing model it outperforms the other methods. For larger sample sizes the coverage results improve, as well. Interestingly the relative half width of the confidence intervals does not decrease with the sample size as much as it does for the other methods.

The method of regeneration cycles has only been used for the queueing model. The results appear slightly worse than the other methods with the exception of the classical treatment. Further enhancements are possible by an extension to a technique called *jackknifing* (see [LK84]). Law and Kelton observed that even with this enhancement the method keeps being inferior

for practical purposes.

Conclusion

For a successful deployment of these methods in quality control charts the subsequent conclusions can be made:

- With the existence of autocorrelation, see section 11.1.5, the σ^2/N estimation scheme neglecting autocorrelation is not applicable which can be seen by the experimental results in table 11.4.
- Regeneration cycles have the strongest theoretical foundations, but the use for quality control charts seems to be not practical. Even if an observed process can be shown to be regenerative, the problem of identifying regeneration points from the transactions of the shop-floor-control system remains.
- The method by Heidelberger and Welch provides the best results of all methods studied. It loses its strength on uncorrelated data, though. Computationally the most effort must be spent on the generation of the periodogram. Using a fast Fourier transformation this can be accomplished in $O(n\log(n))$ steps if n is the length of the time series. The method is rather complex and uses advanced mathematical concepts which could hinder a successful deployment.
- The classical spectral estimation method performs comparable to the method of batch means. It is computationally and from the mathematical point of view more complex than the latter method.
- The method of batch means is a simple to implement estimation scheme with linear complexity. It provides comparably good results and is only outperformed by the method by Heidelberger and Welch.

Each method applied uses a set of parameters which need to be chosen. These are the batch size, the number of autocovariance estimates used in the spectral estimator, or the parameters of the regression technique, respectively. The choice of parameters and the autocovariance structure of the process for which control charts are to be generated should be carefully analyzed prior to the deployment of any method.

For the quality control charts presented in this chapter the method of batch means is chosen primarily for the computational complexity and ease

of implementation on the statistical system used. The size of batches L has to be chosen on basis of an analysis of the covariance structure of the underlying process, see section 11.1.5. It has been set to $L = 150 > 4k_0$ where k_0 denotes the minimum lag for which $\rho(k) \approx 0$ for all $k > k_0$ can be assumed (see [Lav83b]). The value of k_0 could be identified to be not greater than 30 (compare figure 11.11).

11.2.5. Control Charts for Waiting Time

The quality control chart which has been introduced in the last sections dealt with an input parameter of the simulation model, the process time. This section introduces a quality control chart for the waiting time which is an output of the simulation model.

As a chart comparing simulated performance measures against observed characteristics this type of chart has different purposes as compared to charts for input parameters. If a validated simulation model is used, its main purpose is the comparison of planned and realized characteristics of the production line in order to analyze possible problems. If the simulation model has not yet been fully validated, another purpose is the support of the validation process (see section 11.4).

The Importance of Waiting Time

The simulation yields various performance measures for which control charts could be constructed. The waiting time at a work center has some interesting characteristics that make it appealing as a prototype for other performance measures. The following reasons explain why the waiting time is at least as important as the other calculated performance measures:

- The waiting time is measured and calculated on work center level. It is generally easier to maintain charts on this more aggregated level compared to the process step level. The number of possible charts is decreased while for each chart more observations are available.
- The waiting time can neither be further partitioned, nor does it include any input parameters, e.g. the lead time at a work center in contrast (see figure 11.12) includes the waiting time and the process/completion time.

- From the available data waiting time charts can be constructed for any work center, independent of the service type (sequential or parallel). This is not the case for process time charts and therefore also for lead time charts.
- The waiting time directly shows the effect of stochastic influences. Moreover, it should be the ultimate goal to decrease this type of time to zero as no value-add is gained.
- Line profiles (see figure 8.11, for example) result in the insight that lead time and work-in-process are much more sensitive to possible performance problems than the utilization itself as for high work center utilization a strong increase in lead time and work-in-process (mainly caused by high waiting times/queue lengths) can be observed. Martin [Mar98] discusses the advantages of short cycle time manufacturing (SCM) based on the control of lead times compared to continuous flow manufacturing (CFM) which is based on the control of the flow and therefore also of the utilization.
- Measuring the waiting time from the available data is easier than calculating the corresponding measures expressed in parts like queue length or work-in-process because of the present structure of the shop-floor-control database which is rather part-oriented. Having no data about the size of the queue length at the begin of period it is not possible to calculate the queue length by observing the changes over time.

Calculating Observations

To generate quality control charts for waiting times the individual waiting times of parts/batches at each work center to be analyzed need to be calculated from the available data. Basically this is a similar task as the calculation of process/completion times (see figure 11.12). The main difference is that the calculation of the waiting time involves claim processes at different work centers, not just the work center that is to be analyzed.

The calculation involves the following steps:

1. Analogously to the algorithm presented on page 389 the transaction records for the requested work centers are loaded from the shop-floor-control system. Actually, the charts for the waiting times are just an

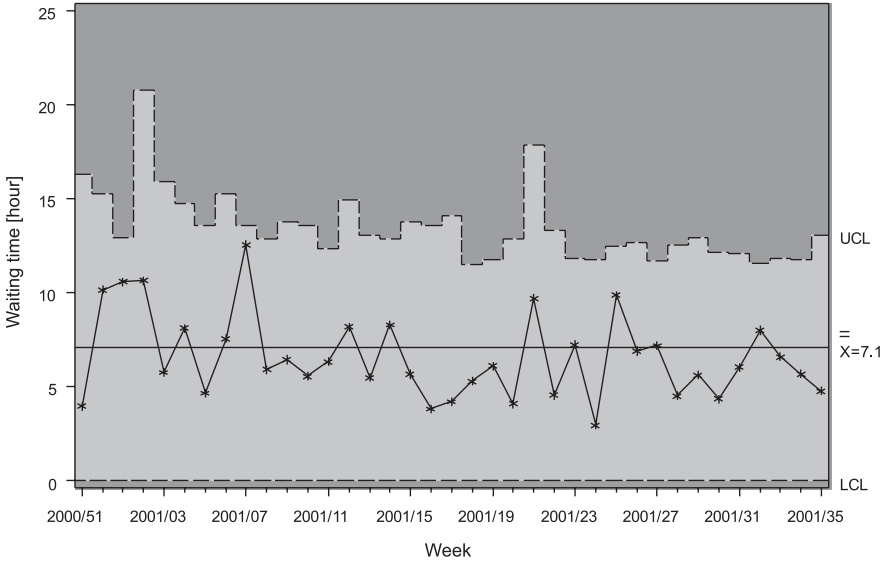


Figure 11.15.: Quality control chart of the waiting time at a sputter machine. The lower control limit is set to zero.

extension of the process time charts, thus the first four steps of the algorithms presented are utilized.

2. To identify the predecessor process steps the simulation model is analyzed: By using the routing table all operation identifiers of preceding process steps are collected. Then a where-clause of the identifiers is created which is used in another query against the shop-floor-control system. This yields a table with all possible transactions at preceding work centers.
3. The two tables containing the in-claim information at the work centers to be analyzed and the out-claims at the predecessor work centers are joined. The result is a table which contains these two times whose difference are the waiting times²⁰.

²⁰The transportation time between different work centers is neglected and modeled as part of the waiting time.

Due to rework some parts might pass a process step twice. Therefore it has to be assured that the corresponding records are joined.

Problems

The calculation of the observations has to encompass some problems, e. g. not every operation that is part of the process flow used by the shop-floor-control system is really physically carried out. This often stems from identity between two or more production lines, i. e. all production lines producing the same products use the same process flow. If a specific process step is not carried out in one of the production lines, the in and out-claim procedures of the shop-floor-control system still have to be carried out. This is normally done just before parts are claimed into process. Thus, the waiting time which can be observed from the shop-floor-control system at these process steps normally is approximately zero.

To bypass this problem first the claim-through operations have to be identified. Then a table with the operation identifiers and the identifiers of their predecessors is created. In step two the predecessor identifiers of the claim-through operations are replaced by the correct identifiers.

Another problem encountered is once again autocorrelation. In contrast to the process time autocorrelation can easily be explained in the observed process: If the n th part that arrives at a work center has to wait longer than average, it is very likely that the $(n+1)$ th part also has to wait longer. Thus, the same measures proposed for the process time charts are also utilized for the waiting time charts:

- Waiting times of batches instead of parts are measured.
- The method of batch means is applied to estimate the variance of the sample mean.

Choosing the Simulation Scenario

Simulating a production line bears the possibility of evaluating different scenarios. If target values for performance measures are derived from the output of the simulated model, the question arises which scenario is to be simulated. When calculating a forecast this seems obvious: The model which best represents the current state of the production is then chosen to achieve a forecast that comes as close to reality as possible (see section 11.3). This is

not the case when production goals are to be derived because a goal is defined to be the value that could be reached within the current circumstances. To create this scenario first of all the current work-in-process is read from the shop-floor-control system. From this the product mix is calculated and the simulation is initialized with it. To set the demand in the model an overall demand goal is used. This value can be set in the simulation request. For this purpose the volume plan containing this figure differentiated for every week could also be used. Volume plans are not production schedules, though, i. e. a volume plan might not be feasible, it might contain strong fluctuations in volumes which cannot be realized by the production facilities. Moreover, a different simulation model for each week could be calculated yielding a possibly different target value. While it would be possible to draw charts having moving target values, just one simulation for the target of the current week is generated. This simplifies both the generation of the charts and the interpretation of them. This is also motivated by the fact that the performance of the current week is most important. The reason why other measured values are shown in the charts is the possibility of evaluating the history of the analyzed processes.

11.2.6. Other Logistical Processes

Numerous logistical processes can be identified in production and corresponding material flow simulations. For most of these processes quality control charts can be envisioned. Not all processes presented can be modeled with the technique presented, though. It is also not worthwhile generating charts for all of these measures as correlations between some processes exist, e. g. lead time and work-in-process are correlated by Little's law. Some of the charts are rather sophisticated and one would implement these after skill has been developed in the use of more basic charts.

One chart for input parameters has already been proposed. In the following the other input processes are listed together with a discussion of their importance and possible problems hindering design and implementation of control charts:

- *MTBF and MTTR*. It is important to control the reliability of key work centers. This can be achieved by the control of both mean time to repair (MTTR) and mean time between failures (MTBF). The base processes are different from the waiting or process time, though. Even

for complex manufacturing processes it is likely that the number of machine outages is small compared to the number of parts produced. This results in low sample sizes or no observations at all (after all it is obviously not in the company's interest to have large samples of MTTR times). One therefore has to decide between fast but less sensitive detection of changes or more sensitive but slower detection. Another problem is that MTTR and MTBF are not normally distributed (the simulation assumes MTBF to be exponentially distributed).

- *Variation of down time.* The variation of the down time has a clear effect on the performance measures like work-in-process or lead time. Instead of creating a chart of its own for this parameter it should be combined with the down time process in an \bar{X}/s chart.
- *Batch size at process steps.* A control chart for the batch size is not reasonable in the current system. Rather than controlling the observed batch size which is not treated as a random variable in the analytical system used, an estimate should be calculated from the shop-floor-control system. This could then — depending on the scenario and goal of the simulation — be used as an alternative to the maximum, minimum, or locally optimized (see section 11.4.1) batch size.
- *Variation of process time.* The squared coefficient of variation of the process/completion time directly influences the calculation of the expected queue length (see equation 3.39 on page 90). It could be controlled in form of the variance of the completion time in an \bar{X}/s chart at process step level.
- *Routing probability.* In section 10.3 the optimization of routing probabilities is presented. The optimal value could be compared to the realized proportions observed on the shop floor.

On the output side of the simulation various performance measures could be controlled:

- *Lead time at work center or process step level.* The lead time is the sum of completion time and waiting time, for both measures the corresponding charts have been developed. The waiting time is a clearly more sensible indicator for the detraction of the production lines's performance.

- *Queue length at work center or process step level.* Through Little's law this measure is correlated to the waiting time. Depending on the type of available information from shop-floor-control systems either the waiting time or the queue length needs to be controlled.
- *Work-in-process.* The work-in-process at a work center is correlated to the lead time. Moreover, it is made up from queue length and parts being processed. Both of these measures could be controlled separately. The queue length, the obvious detractor to the production line's overall performance, is biased by parts being processed. One should therefore concentrate on controlling the queue length or the waiting time and the utilization of work centers.
- *Variation of inter-arrival time.* In equation (3.39) the coefficient of variation of the inter-arrival time shows to be one of the driving forces for performance degradation. Comparing the variance of the inter-arrival time in a \bar{X}/s chart offers insight in the validation of the simulation model and the possible manufacturing problems.
- *Utilization of a work center.* The utilization of a work center is an important measure in capacity planning. It has the disadvantage that it cannot be used directly for problem detection purposes. A low utilization of a work center does not necessarily indicate a problem at this specific work center, e. g. if no parts are available to be processed the utilization of the work center degrades though a problem might have been occurred in the preceding process flow. Due to the highly re-entrant nature of the wafer manufacturing process the detection of this problem is not direct and often complicated.
- *Throughput.* Controlling the throughput of a work center or process step contains the same problem as described for the utilization. Moreover, the simulation yields a throughput for the stationary phase. A problem is created by large lead times compared to fast changing demands and product mixes.

11.2.7. Design of a Decision Support System

For a successful utilization of logistical process control not only charts have to be generated, but a decision support system has to be created. This involves

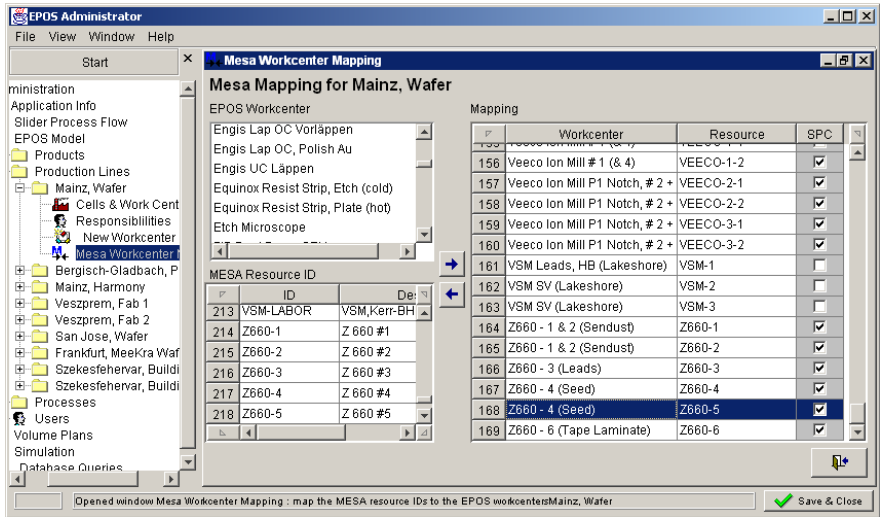


Figure 11.16.: Mapping shop-floor-control machines to the EPOS work centers. A mark in the column SPC indicates that control charts for this machine are to be created.

decisions about which parameters, for which processes are to be controlled or how control charts can be integrated into a company’s reporting site.

Models of complex manufacturing processes can easily contain hundreds of work centers and many thousand process steps. For each of these process steps a quality control chart for the process time and for each work center charts for waiting time or throughput, etc. could be constructed. It seems obvious that this mass of charts cannot be maintained and controlled anymore. According to Ryan [Rya89, p. 76] control charts should be used where trouble is likely to occur. When control charts are first implemented it is also important that they are used where the potential for cost reduction is substantial. It is therefore not important to monitor each process, but select the important processes. This task has to be addressed to the simulation expert of the planning team. The system supports the user by offering two basic measures:

- The work centers for which charts are to be created are limited to the

11.2. LOGISTICAL PROCESS CONTROL (LPC)

WORKCENTER	Operation	Week			
		2001/36	2001/35	2001/34	2001/33
Workcenter A	0344	✓	✓	✓	✓
	0345		✓	✓	✓
	4530		✓		
	5160			✓	✓
	5730	✓	✓	✓	✓
	5770		✓	✓	✓
	6970			✗↑	
	8806	✓	✓	✓	✓
	0960		✗↑	✗↑	✗↑
	2180	✓	✓	✓	✓
	2680				✓
	2750	✓	✓	✓	✓
	2790	✓	✓	✓	
	3565				✓
	3632	✓	✓	✓	✓
	4474		✓	✓	
Workcenter B	0140			✗↑	✓
	9300		✓	✓	✓
Workcenter C	0260	✓	✓	✓	✗↑
	0436		✓	✓	✓
	3460	✓	✓	✓	✓
	3674			✓	
	3782	✓			✓
	3795	✓	✓	✓	✓
	3875	✓	✓	✓	✓
	3960			✓	
	4110		✓	✓	✓
	8520	✓	✓	✓	✓
Workcenter D	9130			✓	
	0295	✓	✓	✓	✓
	0310		✓		✓
	2320	✓	✓	✓	
	2470	✓	✓		
	2700	✓	✓	✓	
	2725	✓			

Figure 11.17.: Interactive overview table for process time charts as it appears in a WWW browser. For each process step analyzed a row shows the state of the last four weeks using different icons. The complete charts can be accessed directly by clicking on the process. Associated waiting time charts are accessible by clicking on the work center name.

important work centers in terms of capacity, throughput, and cycle time. These work centers can be selected by simply checking a box in the EPOS Administrator (see figure 11.16). The creation of charts is then limited to the chosen work centers.

- An overview chart shows a table of all work centers for which charts have been created (see figure 11.17). By use of colored icons it is visualized which charts showed problem situation. All chart can easily be accessed from this overview table by simply clicking on the desired process. This greatly facilitates the detection of current problems in the production line.

Using SAS ODS, HTML files containing references to GIF files are created. Together with the overview tables these files are copied onto an HTTP server. The styles of tables and charts are changed to the standards of the EPOS reporting web-sites into which the control charts have been integrated. Once a week a scheduler starts the SAS programs which are needed to create the control charts.

11.2.8. Results

In the course of this section the background for logistical process control and an implementation of quality control charts of two important logistical processes — one on the input side, another one on the output side of the simulation — have been presented. The generation of control charts is totally automated, selection of charts to be created is just a matter of a few mouse-clicks and has been integrated in the EPOS Administrator.

The generation of charts is accomplished by means of an SAS script running on an SAS server on a scheduled basis. Computationally the main effort is spent on the data selection from the shop-floor-control system followed by the calculation of process/waiting time observations. The actual generation of control charts including calculation of control limits just needs a small fraction of the overall time. To quantify this, process and waiting time charts have been created for seven selected work centers over a time period of 260 days. The generation of waiting time charts for these seven work center and process time charts for 33 process steps being performed on them took 31.6 minutes on a Pentium 3, 800 MHz, 256 MB Ram. 31.6% of this time has been spent to load the transaction data for the specified work centers from the shop-floor-control system. Another 27.0% was needed to

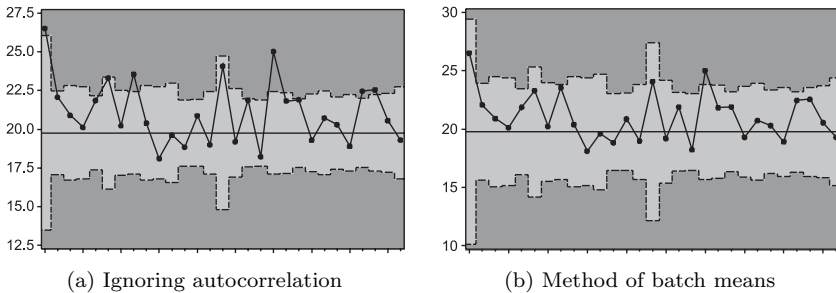


Figure 11.18.: The effect of autocorrelation. Both charts show the a quality control chart of a the overcoat sputter process step. The left chart has been created neglecting the effects of autocorrelation on the variance of the sample mean. To create the right chart the method of batch means has been used.

load data of the preceding work centers. Combining the data accounted for 34.0% of the overall time. For the process time charts calculation of control limits, drawing charts, publishing in HTML and GIF format, etc. only 2.2% were needed.

Coping with Autocorrelation

The analysis of the covariance structure within logistical processes revealed that autocorrelation cannot be neglected. The method of batch means is a simple and inexpensive method which can be employed to solve the problem. Figure 11.18 shows the effect of autocorrelation on the calculation of control limits of quality control charts. To create the chart in figure 11.18.a the variance of the sample mean has been estimated ignoring autocorrelation in the process. Though the process is in control the charts show some samples that fall out of the acceptable region. Figure 11.18.b shows the chart for the same process, the estimation of the sample mean has been carried out using the method of batch means, though. This time nearly all samples fall within the acceptable region.

Comparing these charts, the effect of autocorrelation becomes visible. As individual observations in the sample are not independent the information gained by each observation is not comparable to that gained by an independent observation. The result is a misleading standard error of the sample

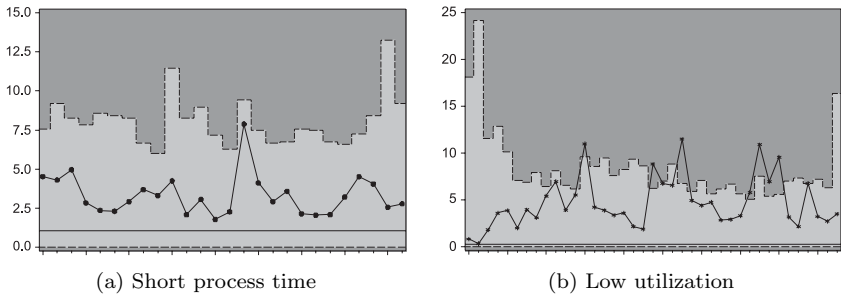


Figure 11.19.: Process time (a) and waiting time (b) chart with non-matching targets.

mean. As an effect the control limits are set too narrow.

General Discussion of Quality Control Charts for Logistical Processes

The following points discuss quality control charts for process and waiting time:

- The targets for process and waiting time charts are taken from the simulation model, the variance is estimated from observed data. If charts are to be created without further user interaction this could become a problem. Usually, if no target is available, the variance is estimated from data taken from a controlled phase before the actual deployment of the control chart. During this phase it has to be assured that the process is in statistical control which is only possible by manual interaction.

While it is not possible to automatically judge whether a process is in control or not, statistical methods are available for data cleansing, e. g. tests for outliers like the ones by David-Hartley-Pearson or Grubbs (see [Har93]) can be used to eliminate outliers from data sets used to calculate control limits.

- Currently control charts are available by means of static HTML pages which are repeatedly updated. Interactive control charts would allow a wider range of more up-to-date charts. This is hindered by the large computational effort needed to select and calculate observations from

the shop-floor-control system. This problem could be circumvented by the use of summary tables.

- Figure 11.19.a shows a chart for a sputter operation with a short process time. The parallel machine takes up to nine parts and processes them from 20 to 45 minutes depending on the operation. The variation of the measured completion time is with 2.22 quite high. Combined with autocorrelation in the observed data this leads to broad control limits around the target time. The lower control limit is set to zero in cases where the formula yields a limit below zero in order to not confuse the users. The chart in figure 11.19.a fails to trigger an alarm, though it is apparent that the process does not perform at the specified target level. By the use of run rules as presented by Western Electric [S⁺56], alarms would be triggered in this situation. The chart presented would fail the test for *eight points in a row on one side of the centerline*, for example.

The specified process time is machine and capacity-oriented, i.e. the machines at this work center could be run at the specified speed, if it was a bottleneck. Otherwise all operator handling and activities that could possibly be carried out in parallel are part of the observed process time.

Generally, it appears that targeted and observed times fit the better the longer the processes last.

- For some work centers a very low waiting time is simulated (this might be the case at work centers having batch size one and a low utilization). Figure 11.19.b shows a waiting time chart for such a work center. Obviously the process is not in control at the simulated level. This has basically two reasons: (i) the observed waiting time includes the transportation time and (ii) at non-bottleneck work centers an operator is not always immediately available to process the just arrived parts. These effects lead to an actual waiting time which is likely to differ from zero.
- The waiting time simulated for work centers having large batch sizes sometimes exceeded the observed waiting time. This problem can be solved by the calculation of locally optimized batch sizes, see section 11.4.1.

- Compared to approaches which detect changes from current performance like *Days Added* by Bonal et al [B⁺01b], control charts have the great advantage to be target/actual comparisons. This enables the detection of potential and possible performance gains.

In general the created charts have been very valuable for model validation purposes (see section 11.4). Before a control chart can be put into production, i. e. before it can reasonably be used for problem detection by line control personnel it should be thoroughly checked and validated, though.

11.3. Work-in-Process Forecast

A daily task in line control is to forecast the current work-in-process and match the projection with the production schedule. This is done to evaluate the current status of the production and detect delayed parts for which actions could be initiated. This task is normally done using spreadsheets which often involves tedious manual work. Even if it is automated, systems often neglect stochastic influences of the production. In the following an automated forecast for the current work-in-process is presented which makes use of the analytical performance evaluation. Differences compared to discrete event simulations are discussed.

11.3.1. Calculating the Forecast

Forecasting the outcome of the current work-in-process involves various tasks which can be identified as set-up of the simulation model, matching the actual parts in the production line into the simulation model, and reading the simulating outcome of each part.

Set-up of the Simulation Model

For generating projections on the operational level the simulation model has to be set up differently as compared to the model on the tactical level. The main purpose of the model on the operational level is to reflect reality as close as possible. This implies that the tactical model has to be adjusted to the current situation. A key performance measure which applies to every work station in the model is the utilization. It directly influences performance measures like work-in-process and lead time and is itself strongly influenced

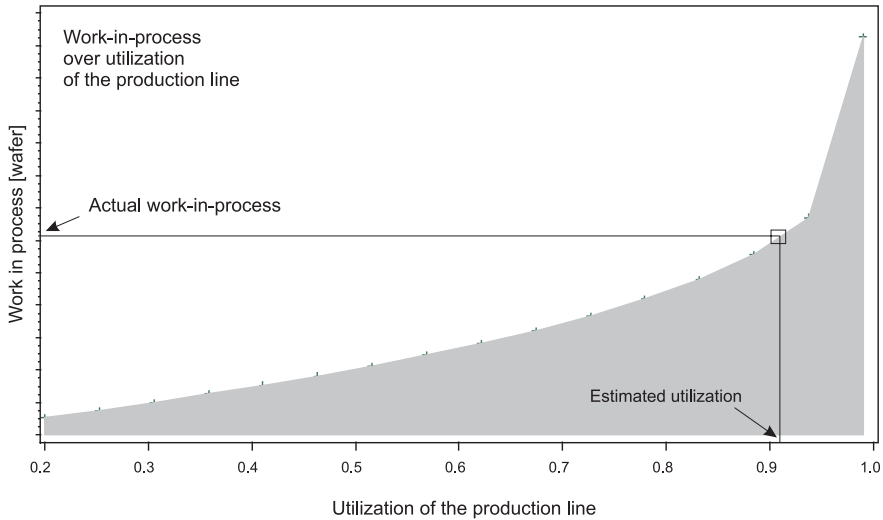


Figure 11.20.: Estimation of the current utilization in the production line

by the flow of parts through the production line which is correlated to the demand. Without changing the tactical parameters of the model it is therefore possible to adjust the utilization by changing the demand in the model.

Figure 11.20 shows the expected work-in-process over the utilization. The current work-in-process can be read from the shop-floor-control system. It is now the task to find the utilization that would cause such a level of work-in-process. The expected work-in-process can be thought of as a strong monotonic function of the utilization. Thus, a binary search can be applied to find the utilization for a given work-in-process level making use of the fast calculation of the analytical simulation engine. Adjusting the demand to reach a goal work-in-process within less than one percent relative error of the Mainz wafer model takes less than 30 seconds.

When adjusting the demand the current product mix has to be reflected in the simulation model. This implies the use of mapping tables between simulated product groups and product types in the shop-floor-control system. Moreover, it is important to select just the parts which are actually processed, i.e. all parts which are on hold, in ship buffers, marked as experiments, etc. have to be excluded.

MATCHING OUTCOME AND DEMAND

Input: parts of current work-in-process with expected outcome, volume plan

Output: parts of current work-in-process with demanded week

```

begin
  for each product
    sort listOfParts of product by expectedDate
    for week = week(today) to lastWeek
      sum = 0
      quantity = quantity(volumePlan, product, week)
      while (sum ≤ quantity)
        part = nextElement(listOfParts)
        sum = sum + goodProb(part) // parts are yielded
        part.demandedWeek = week
        store part
      end
    next week
  next product
end

```

Figure 11.21.: Matching parts into the demanded weeks by using the expected date of outcome (*expectedDate*)

Adjusting the model could also incorporate estimation of further parameters from the shop-floor-control system. To be able to judge the outcome of the forecast the model has to be known, though. Unattended changes in parameters before calculating the forecast would change the model in an uncontrollable fashion which would hinder the ability to judge the outcome. A possibility of integrating estimations from shop-floor-control systems would be a different set of operational parameters. The current implementation just supports the adjustment of the utilization.

Matching the Parts

After the model has been set up, parts which can be read from the shop-floor-control system have to be matched into the simulation model. A part is identified by its product type, current operation in the process flow, and machine that it is currently processed or queued at. On the model side

these three attributes are reflected by the process step, see section 4.3.1. The effort of maintaining three mapping tables can be facilitated by the matching algorithm. The most important attribute when calculating a forecast is the operation for which a mapping algorithm exists between EPOS and the shop-floor-control system's identifiers. Product types which have not been mapped to product groups are set to the default product group which is calculated as the product group with the largest product mix. If a machine cannot be mapped to a work center, it is randomly assigned to one which performs the operation for the specified product group.

In contrast to the set-up of the simulation model all parts have to be matched into the model. This includes parts which are stored in ship buffers, i. e. parts which do not increase the load of the production line in order to be completed, but which are available to be shipped.

Simulating the Expected Date of Completion

After the simulation each process step is associated with a probability that the part will leave the production line as a good part, see section 3.7. Moreover, the lead time from that process step to the last process step in the flow for good parts is also known, see section 3.17. It is then just a matter of reading these values from the simulation model and associating them with the matched parts. The information for each matched part is written back into the EPOS database. The expected date of outcome is set as the current time-stamp plus the lead time for good parts.

11.3.2. Matching Outcome and Demand

To be able to identify possible problems in the current line loading the expected outcome of a part has to be set in relation to the demanded date which can be taken from the volume plan. An algorithm which matches individual parts into the volume plan and sets the demanded week has been developed (see figure 11.21). By comparing demanded and expected week a difference showing the lateness which can be positive, zero, or negative is obtained.

11.3.3. Presentation

Parts of the current work-in-process have been assigned an expected date of outcome and the week that the part is demanded. The information is stored

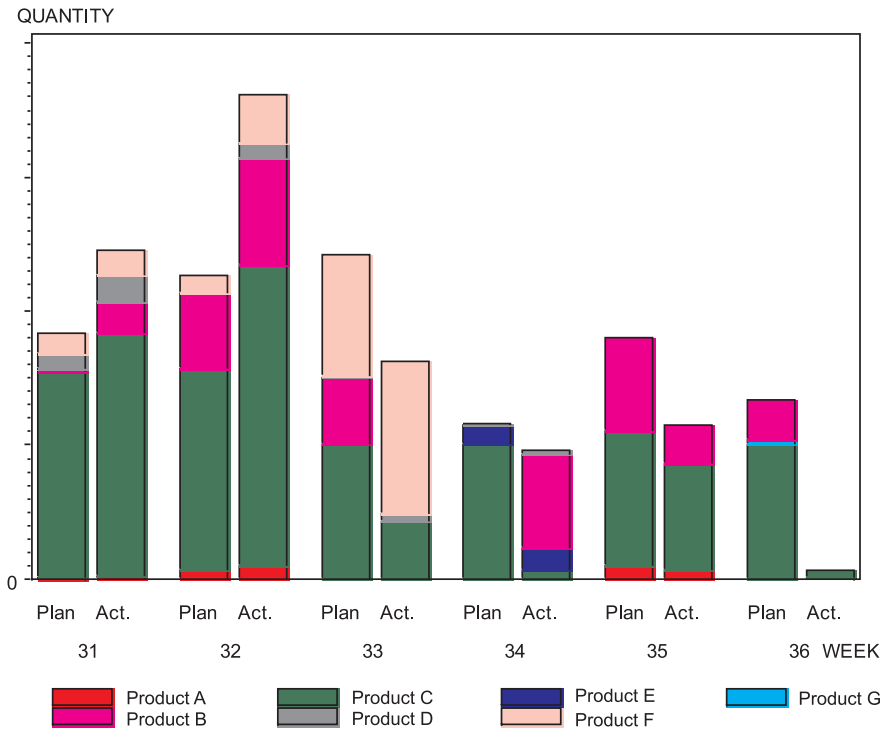


Figure 11.22.: Expected output versus plan

in the EPOS database and needs to be published and distributed. The data needs to be presented in different ways and on different levels of aggregation depending on the need of the line control department. Exemplary, a chart and table that both have been created using SAS are shown:

- Figure 11.22 shows a stacked bar chart. For each future week of the volume plan two bars — one for the planned volume, another for the expected yielded outcome of the current work-in-process — are shown. Different product types are stacked within each bar. This chart is used as a high level overview giving an impression of the current work-in-process distribution. The creation does not require the match between expected and demanded week.

	TYPE			
	Bottom		Top	
	Quantity	Mean	Quantity	Mean
		Delta		Delta
Operation				
0060 Release			8	
0110			32	
0550	8		32	
0580	1			
0890			8	
0960			1	
1065	17		32	
1095	11		1	
1105	10	1	2	
1155			7	
1160	3	1		
1370	2	2		
1585	1	2		
1930	2	2		
2140	3	2		
3510			4	
3725			1	
3760	3	2	4	
3800			1	
4020	2	1	1	
4230			6	
4945	3	1		
5810			1	
7700	4	1	2	
8789			2	
9850 Final Visual Inspection	15	0	12	0

Figure 11.23.: Quantity of parts and average lateness. The first column shows the process flow at which work-in-process is reported, the second and fourth columns present the number of parts at each operation. The columns titled *mean delta* show the average difference between expected and demanded date of shipment in weeks (see further description on page 432).

- The table in figure 11.23 is an example of a report which makes use of the matched expected/demanded date of finish. The report contains a table for each product group showing the operations of the process flow where work-in-process is present in the first column. It is sorted by the flow of material, starting with the first operation in the first row. The table is read bottom-up, i.e. one starts with the last operation, 9850 **Visual Inspection**. There are 15 bottom and 12 top parts at this operation. The average lateness of these parts is both zero meaning that there is a demand and these parts are expected to be finished in time to meet that demand. At operation 7700 four bottom parts are on average one week behind schedule which is indicated by the figure in the red cell. For top parts the situation is different: at operation 8789 two parts do not have any demand shown by the missing average lateness.

11.3.4. Results

The work-in-process forecast has been implemented in the model generator and runs scheduled multiple times per day. Reports implemented as SAS programs use the results and create charts and tables which are published within the EPOS web-site. The process is totally automated and uses the simulation model which is maintained within the process of *integrated simulation*.

According to line control that currently validates the created reports the results are promising. This may come as a surprise; after all, the analytical simulation model is not the natural choice for this task. A design of such a model would include the set-up state of all machines, all current machine failures and their history, current operator availability, etc. Compared to the analytical system used this would involve much higher effort to maintain the more detailed model and a tighter integration of shop-floor-control, maintenance systems, and simulation. Moreover, the complex stochastic behavior of the production line implies the need of great amount of computational power to gain statistically significant results because the transient phase would have to be simulated by many independent replications.

11.4. Model Validation

One aspect²¹ of validation lies in deciding whether the model is an adequate representation of the real world that it is intended to describe (see [GH74, p.412]). If historical data of the actual system is available, it can be used as input to the model and the subsequent model output can be checked with the actual outcome. The shop-floor-control system is the place where actual data is stored. It is therefore the task to integrate shop-floor-control and simulation in order to enable a systematic validation of the simulated models.

It is important to realize that statistical methods have to be used when comparing data from shop-floor-control to observed data as the latter is a realization of a stochastic process. LPC is a system based on the same assumptions and therefore a means to systemize the validation process.

11.4.1. Locally Optimized Batch Size

The quality control charts of process and waiting time presented in section 11.2 have been used to facilitate the validation process of the model of the Mainz wafer line. An analysis of the waiting time at work center level revealed that the current analytical model needed modifications concerning the batch sizes used in the model.

It could be shown that work centers with very large batch size like ovens did not experience the simulated waiting time. For capacity analysis the maximum batch size has been set in the simulation model. This procedure assures maximum capacity at each work center. Due to the (b, b) rule of the simulation engine a large batch size can drastically increase the number of parts waiting to be processed in situations of low utilization. This model is not consistent with reality. Ovens, for example, often have a minimum batch size which can be substantially lower than the maximum batch size. As an analytical treatment of the (a, b) rule is not yet available an optimal load size concerning the queue length or waiting time should be set in the simulation model.

²¹besides the internal correctness of the model

Deriving a Local Approximation

In section 3.14 an approximation for the queue length in front of a work center is presented. Equation (3.38) and (3.39) are now thought of as a function of the batch size b , i.e.

$$E[Q](b) = \frac{b-1}{2} + \frac{\varrho}{1-\varrho} \frac{bP_c(C^2[I^b] + C^2[C])}{2} \quad (11.77)$$

with P_c as the probability that all c servers are busy given by

$$P_c = \frac{\frac{(c\varrho)^c}{c!} \frac{1}{1-\varrho}}{\sum_{i=0}^{c-1} \frac{(c\varrho)^i}{i!} + \frac{(c\varrho)^c}{c!} \frac{1}{1-\varrho}}. \quad (11.78)$$

For the single server scenario ($c = 1$) the optimal batch size in terms of the expected queue length can be calculated. As many other performance measures like waiting time, work-in-process, lead time, etc. are directly related to the mean queue length (see chapter 3) an optimization of the queue length optimizes the performance measures mentioned, as well.

It can easily be shown that for the single server case P_1 is equal to ϱ because the probability that the server is busy is obviously equal to ϱ . The total utilization ϱ^{tot} (see equation (3.36)) depends on the batch size and with $u = \lambda^X E[X] E[C]$ it can be written as $\varrho^{tot} = u/b$. Setting $v = C^2[I^b] + C^2[C]$ yields the simplified form of (11.77) as

$$E[Q](b) = \frac{b-1}{2} + \frac{u^2 v}{2(b-u)}. \quad (11.79)$$

Figure 11.24 shows a plot of $E[Q](b)$ over the batch size b treated as a real number in this case. The function is then defined on $]u, \infty]$. If b becomes less than u , $\varrho = u/b$ becomes greater than one which is not permitted for the stationary state of the queuing system. On the right side of u the mean queue length shows two different behaviors: Very close to u the utilization ϱ increases and function (11.77) is strongly influenced by the second part of the sum, mainly by $\varrho/(1-\varrho)$. For larger batch sizes the first part of the sum has much greater influence. Between these two parts the function of the expected queue length has exactly one minimum which can be found by calculating the first derivative and setting it to zero:

$$E[Q](b)' = \frac{1}{2} - \frac{u^2 v}{2(b-u)^2} \stackrel{!}{=} 0. \quad (11.80)$$

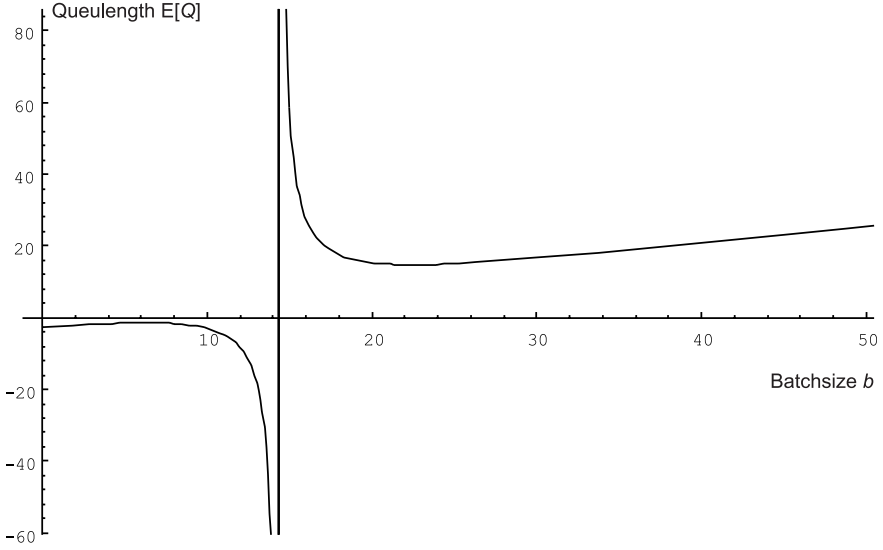


Figure 11.24.: Mean queue length as a function of batch size.

This leads to the quadratic equation

$$b^2 - 2ub + u^2(1 - v) = 0 \quad (11.81)$$

which has the two solutions $b_{1,2} = u \pm u\sqrt{v}$. The smaller one, $b_1 = u - u\sqrt{v}$ is not valid because the utilization $\varrho = u/b_1 = u/(u - \sqrt{v})$ would become greater than one. This leaves the optimal batch size $b^* = u + u\sqrt{v}$. The second derivation of the expected queue length can be written as $E[Q](b)'' = u^2v/(b - u)^2$. Setting $b = u + u\sqrt{v}$ results in

$$E[Q](b)''|_{b=u+u\sqrt{v}} = \frac{u^2v}{(u + u\sqrt{v} - u)^3} = \frac{1}{uv^{3/2}} > 0 \quad (11.82)$$

proving that the mean queue length has a minimum at $b^* = u + u\sqrt{v}$.

The queuing model presented in chapter 3 treats the batch size as a mathematical integer. To be able to use the locally optimal batch size in the model is has to be rounded in some way. It was chosen to use the ceiling function to prohibit a too small batch size which could possibly result in an

overload situation. Furthermore the squared coefficient of the input batch size has to be substituted using the approximation [Zis99]

$$C^2[I^b] \approx \frac{E[X]}{b} (C^2[X] + C^2[I^X]). \quad (11.83)$$

This finally leads to

$$b^* = \left\lceil \lambda^X E[X] E[C] \left(1 + \sqrt{\frac{E[X]}{b} (C^2[X] + C^2[I^X])} \right) \right\rceil. \quad (11.84)$$

Due to the complicated structure of P_c (see formula (11.78)) an approximation in the multi-server scenario cannot be derived as easily, because P_c depends on the total utilization ϱ^{tot} which in turn depends on the batch size b . This complicates the derivation which is needed to calculate the optimum.

It is easy though to search for the locally optimal batch size of the work center by simply calculating the expected queue length $E[Q](b)$ for all possible batch sizes between the minimum and maximum batch size. To simplify this calculation a further approximation to P_c is offered by Bolch [Bol83]. He shows that in case of high utilization ϱ the probability that a part has to wait can be approximated by the simple to calculate formula

$$P_c \approx \frac{\varrho + \varrho^c}{2}. \quad (11.85)$$

Bolch proofs that $\varrho \leq P_c \leq \varrho^c$ for $c = 1, 2, \dots, \infty$. Formula (11.85) is just the arithmetic average of these two limits as for large utilization the exact value of P_c is very close to the arithmetic average of the limits. Bolch shows that for small utilizations the geometric average, namely

$$P_c \approx \varrho^{\frac{c+1}{2}} = \sqrt{\varrho^{c+1}} \quad (11.86)$$

performs better. Using this result it is computationally not costly to run through all possible batch sizes in order to find the minimum of $E[Q](b)$ for multi-server work centers.

Simulation of the Locally Optimum Batch Size

Currently the calculation of the locally optimum batch is carried out in the model generator (see chapter 7). It is available to the end-user as an optional

feature which can be switched on in the administration client (see chapter 9). Being implemented totally transparent all types of simulation modes like line profiles, work-in-process forecast, etc. can make use of it.

The drawback of the current implementation is the need to calculate the performance measures twice. This is caused by the use of intermediate results like the mean input batch size or the arrival rate of batches at a work center which are needed in order to calculate the locally optimum batch size. The first simulation run is thus needed to calculate the intermediate results, then the new batch size is calculated and set in the simulation model. A second run is used to finally calculate the performance measures. The reason for this is the absence of a maximum and minimum batch size in the AMS simulation model (see [Kle00a]).

11.4.2. Batch Size Correction to the Transport Batch Size

Many work centers process parts sequentially, e. g. using a microscope parts are inspected manually one after another. In the tool-parameter-sheets (see chapter 5) the mean process time would probably be specified for a batch size of one part, though parts are normally not inspected using this batch size, but rather with the batch size in which parts arrive at the work center. This so-called transport batch size is influenced by the carriers used to transport parts, the transport batch size at the start of the line, and the procedures used to split off parts in case of scrap or rework. Using the shop-floor-control system it is possible to estimate the mean transport batch size at each step of the process. Figure 11.25 shows an example of such an analysis.

When the mean size of transport batches arriving at each work center is estimated using the shop-floor-control system it can be used during the automatic model generation (see chapter 7) in order to build a simulation model that reflects the transport batch size used in the production line. This involves changing the batch size at the work center which in turn would change the capacity. To change the batch size without affecting the capacity, obviously the mean process time has to be adjusted accordingly, e. g. if parts arrive in front of the aforementioned microscope with a mean batch size of four parts the mean process time must then be set to four times the process time of one part. It is important to note that this can only be done for sequential work centers. Parallel machines in contrast are characterized by a fixed process time which is considered constant over the possible batch

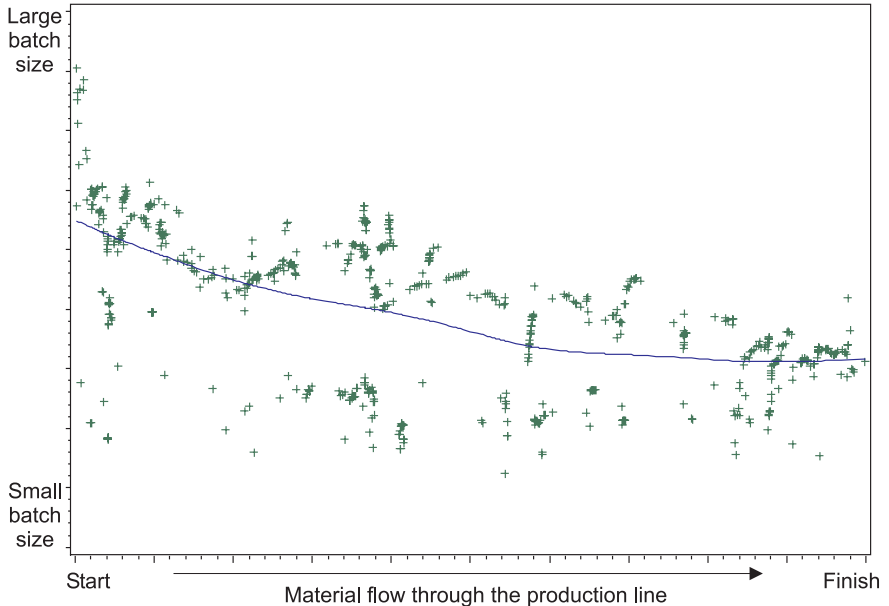


Figure 11.25.: The average batch size over the production line. One can see that it is constantly decreasing due to parts being split off because of rework or scrap. Analyzing the coefficient of variation of the batch size shows that it is rising from the first to the last operation of the line.

sizes²².

The described procedure is implemented in the model generator as an optional feature. As changing the batch size to the transport batch size is only applied for sequential machines this feature is totally orthogonal to the calculation of the locally optimum batch size which only applies to parallel work centers. It is now possible to calculate all scenarios including the work-in-process forecast and line profiles with both correction methods turned on, either one, or none of these features.

²²In general, the mean process time will not exactly be constant even for parallel machines. For the sake of simplicity and easier maintenance of the simulation model a constant cycle time is assumed for parallel machines, see figure 4.4.

Work center	Type	A: Max. load size			B: Corr. load size			Delta
		b	ϱ	$E[Q]$	b	ϱ	$E[Q]$	
Oven	parallel	108	0.17	53.92	29	0.66	18.94	-64.9%
Spray etch	parallel	24	0.11	11.73	12	0.23	6.23	-46.9%
Insulation bake	parallel	14	0.22	7.29	8	0.43	5.44	-25.4%
Sputter	parallel	9	0.10	4.60	3	0.46	1.15	-75.0%
Microscope 1	sequential	12	0.40	8.53	4	0.40	3.77	-55.8%
Microscope 2	sequential	8	0.55	14.28	4	0.55	11.27	-21.0%
Stepper	sequential	1	0.46	0.65	4	0.46	2.18	+235.3%
Tester	sequential	1	0.59	1.50	4	0.59	3.70	+146.6%

Table 11.5.: Comparison of simulation results using maximum load size against using locally optimized and transport batch sizes.

11.4.3. Results

To evaluate the differences between possible batch sizes, simulations of the model of the Mainz wafer production line have been performed.

Table 11.5 shows the results for some individual work centers taken from two calculations of the model. Scenario *A* is based on the maximum load size whereas scenario *B* uses locally optimized and transport batch sizes. Note that locally optimized batch sizes are only calculated for parallel work centers whereas the transportation batch size is only applied in case of sequential work centers. For parallel machines utilization ϱ changes in contrast to sequential machines where ϱ has to be kept on the same level by definition. Changing load sizes to transportation batch sizes can be applied in either direction — increasing or decreasing the load size. Locally optimized load sizes are always lower or equal to the maximum load size.

11.5. Conclusion

11.5.1. Summary

This chapter has presented three approaches to integrate analytical performance evaluation with shop-floor-control systems. Four generic strategies can be identified: (i) extraction of model input parameters, (ii) problem detection, (iii) generation of operational plans, and (iv) validation of the simulation model. Extraction of model input parameters involves the analysis of estimators and is widely discussed in the simulated-related literature. Problem detection is triggered by comparing targets and actual performance.

The model of *integrated simulation* enables these comparisons as the permanent availability of current simulation models provides means to derive target values. Moreover, the availability of current models also enables the generation of operational plans. The validation of the simulated models is closely related to problem detection and is therefore facilitated by the same means, i. e. comparing actuals against targets from the simulation model.

Logistical process control has been introduced by the design of quality control charts for two important logistical processes. The following problems occurred:

- Control charts for process/completion time process can only be generated for parallel machines.
- Calculation of observations for the process and the waiting time has to be based on the transaction table and is therefore time consuming.
- The choice of the simulation scenario strongly influences the target values. Great care has to be employed if reasonable targets are to be derived.
- The simulation model and the calculation of actuals is based on different sets of assumptions which sometimes do not match, e.g. the simulation model does not include transportation time, it is contained in the waiting time actuals, though.
- Autocorrelation aggravates the creation control charts, especially for waiting time charts by biasing the estimator for the variance of the sample mean. A possible solution is found in the method of batch means.

In spite of these problems quality control charts for logistical processes can be designed, implemented and maintained. They have clear advantages over methods for problem detection which do not make use of target values. The advantages of the presented system can be summarized as follows:

- After LPC has been deployed hardly any additional maintenance effort is required. The current system makes use of the already supported simulation system and uses the mapping tables for work centers and product types which are already needed for the maintenance of the simulation model.

- Due to the large sample sizes which accumulate during the period of one week the effect of autocorrelation is lessened while increasing significance of the underlying statistical tests.
- A navigational scheme has been designed which allows fast access to all created charts and simultaneously serves as an overview to the current state of each process.
- Logistical process control is based on the well established methodology of statistical process control. The charts can intuitively and easily be interpreted.

If a system for *integrated simulation* is run, LPC is available at very low maintenance costs. It proves to be very valuable in the validation process of the simulation model. Charts for validated processes can easily be put into production and offer a fast and easy possibility of detecting potential manufacturing problems.

A forecast for the current work-in-process has been designed on basis of the analytical simulation model. Though this model is aimed to answer questions of tactical production planning it can successfully be re-used to tackle operational problems. The fast and automatic calculation of the forecast enables its use as an on-line tool. Still, the selection of the right simulation scenario remains crucial for the validity of the forecast. In the current approach the model is adjusted to the current level of work-in-process.

Systematic validation of the simulation model of the wafer line using the control charts from LPC showed that the batch size in the simulation has not been correctly set for all machines. Especially for work centers with very high batch sizes unrealistic queue lengths were predicted as a result from the (b, b) batching rule. An approach to locally optimize the batch size for a given level of utilization which solved the problem of large queue lengths has been developed and implemented.

11.5.2. Review of the Current Use of Shop-Floor-Control Systems

In [Spu92] the primary goals of shop-floor-control systems can be summarized under the following points:

1. *Control and planning of production.* To achieve this overall goal shop-floor-control systems need to track the states of orders and work-in-process, the states of resources, the performance measures of employees, and the use and consumption of materials.
2. *Enforcement of production flow.* Shop-floor-control systems are needed to ensure that parts are processed at the right machine with the right recipe, or as Leachman [LH97] puts it: "make manufacturing mistake-proof".
3. *Cost accounting and control.* Measures of line performance can be used for cost estimation in cost accounting.
4. *Transparent production.* By means of controlling the flow of work-in-process through the line by means of on-line available work-in-process snapshots, shop-floor-control systems enhance the transparency of the production processes. Moreover, by controlling the availability of process equipment, i.e. measuring MTBF and MTTR, the performance of the maintenance processes can be analyzed.

If shop-floor-control systems are solely used for these purposes, the potential power of these system is not fully taken advantage of, i.e. the systems are not systematically used to uncover and identify problems of logistical processes, possibilities of comparing the manufacturing performance to planned parameters and simulated performance measures are not supported. Often the design of shop-floor-control systems is strongly oriented by their primary goals. This implies that a further use is often complicated. In detail the following topics hindering a further utilization can be identified:

- If systems for controlling maintenance parameters are deployed isolated from shop-floor-control systems, their data models need to integrated. Without any integration gathered maintenance data can only be used by the maintenance department. To use the data for capacity planning, simulation, or lead time analyses a tighter integration is required.
- Other control systems used on the shop floor like virtual Kanban systems often make use of the primary shop-floor-control system and demand that some basic objects like operations, sectors, or product types are not modeled in the way proposed by the shop-floor-control system. This leads to increased maintenance costs and creates problems when

trying to integrate the shop-floor-control system with other production management applications like simulation systems.

- The concept of identity between production sites demands process steps being performed at one site to be included in the process flow of all other production sites, as well. These process steps have then to be claimed though resulting in additional operator effort and complications in the analysis of production line efficiency.
- Variabilities of process flows, process times, machine failures, batch sizes, transport lot sizes, etc. are usually not measured directly.

While it is possible to derive some (but not all) of the required variance measures, the calculation from transaction data requires enormous effort which could be saved if these measures were calculated within the available summarization engines.

- While numerous technical parameters are collected in machine logs at many machines, often hardly any logistical information like batch sizes, set-up or monitoring times are available.
- Measures like process time, waiting time, lead time, etc. can relatively easy be calculated from the present data structures. This is not the case for the corresponding measures in terms of parts like queue length or work-in-process as no machine state information is gathered.

11.5.3. Requirements for Future Shop-Floor-Control Systems

The effective use of shop-floor-control system as means to control logistical process and detect possible problems requires certain data to be available. This data needs to be collected and summarized for all important machines in a common structure in order to avoid the necessity of creating different reports for each machine. Moreover, data should be available in a multi-dimensional format to facilitate the analysis on different aggregational levels. This could be a snow star scheme modeling the company's structure like it the case in the EPOS master data structure. For the time dimension, summarizations should be generated at least per shift basis. In detail the following information needs to be made available as either raw data or summarizations:

- Batch size of parts processed together in a batch

- Process time of each processed batch at each process step
- Waiting time of each batch at each process step
- Average queue length at process step level per period
- Average parts in process at machine level per period
- Inter-arrival and inter-departure time between consecutive batches at process step level
- Sample variance of inter-arrival and inter-departure time per period
- Sample variance of the process time per period

The need for extensions to current shop-floor-control systems has been addressed by Leachman [Lea97] who focuses on OEE²³ and capacity analysis, but the information required for control of logistical processes goes beyond of what is previously demanded.

11.5.4. Outlook

Logistical process control has been proven a very valuable tool in the validation of the Mainz wafer model. It is now the task to select important processes, validate these on both the simulation and the actual side, and initiate a weekly review of the processes' performance. The next step would be to implement other important charts like throughput charts.

The work-in-process forecast can replace the to date manual creation of forecasts on spreadsheet basis. This would save much effort and would improve the quality of the predicted dates of outcome. This is due to the possibility of the simulation model to react to the current level of utilization in the production line which is not possible by using spreadsheets.

²³Overall Equipment Efficiency

Chapter 12

Conclusion

12.1. Summary

Analyzing the questions of production planning in today's technology manufacturing processes with respect to the mass of methodologies, models, and literature available in operations research (OR), it becomes obvious that there still exists a practicality gap. On the one hand most of the planning is still done using spreadsheets neglecting any stochastic element in the manufacturing process, on the other hand models have become more and more general, but still, their application is often hindered by mathematical complexity and the amount of parameters needed. While queueing network analysis has become applicable even to batch processing, its practical use has still been limited. This thesis presents an approach to solve this dilemma.

Integrated simulation provides a framework for integrating simulation methods into the production planning processes. The goal of the concept is to make current mathematical methods available to answer the questions of real-world manufacturing processes. Reviewing previous research in both fields — mathematical modeling and production planning — shows that the holistic approach of integrating all phases of production planning, from data acquisition to the distribution of results, has hardly been considered. Especially an environment with focus on queueing network analysis has not been studied before. This thesis develops the cornerstones of *integrated*

simulation, namely the collaborative creation of simulation models, automatic model generation, and the integration of simulation results. Emerging middleware, internet standards, and the computing power of today's work stations make it possible to create and maintain systems for *integrated simulation* in an efficient way even in a rapidly changing environment.

EPOS is a system that implements many of the features required by *integrated simulation*. It is based on object-oriented models on different levels of abstraction. The need for these different types of levels stems from the differences between the models: For data acquisition and representation a model which is closely related to the real world, for the application of mathematical methods a rather abstract model is needed. The general analytical model on the one hand allows its use in many manufacturing scenarios, on the other hand modeling techniques are required to represent reality correctly. Techniques that allow to map complex real-world work centers to queueing sub-networks can be abstracted and thus allow their use during automatic model creation. This makes the advantages of fast analytical performance analysis available while hiding modeling details from end-users. To derive the mathematical model from the real-world model an *intelligent* and robust model generator supporting a parameterized model generation is introduced in this thesis.

Additionally, from another point of view the automatic generation of simulation models proves to be very important: Every application like capacity planning, forecasting work-in-process, product mix and routing optimization, creation of operational target values, etc. requires a simulation model with a different focus. Different scenarios might demand different simulation methodologies (analytical or discrete event). Automatic model generation makes it possible to generate different models which are created to the specific needs of the scenario from the same source of data.

EPOS has been developed at IBM Deutschland Speichersysteme GmbH in Mainz. At the moment all of the production lines for read/write heads in Mainz and Hungary are simulated and their performance measures are supervised. The models of the five production lines currently contain 570 work centers, 1606 operations, 257 products, 15,276 process steps, and 20,757 routings. The models are maintained by 256 persons from engineering, manufacturing, and maintenance.

Throughout the development of EPOS scalability has been an important issue as it clearly influences the range of applicability. The system's range of deployment reaches from the EPOS Analyzer on a laptop computer to

a company-wide use including the simulation of production lines of many thousand objects.

The system performance can be described by the subsequent figures. A current model of the Mainz wafer line — the most complex of the five production lines modeled — contains 21,260 simulation objects (work centers, operations, routings, etc.). On a standard PC¹ the evaluation of all described performance measures (see chapter 3) takes about four seconds. The calculation of a line profile with 20 simulation runs requires about 32 seconds. Generating the simulation model, calculating about 52 weeks including an optimization of the product mix, a line profile, a forecast of the work-in-process, and storing all results takes 1.8 hours on average.

Crucial to the deployment of a system for *integrated simulation* is the user acceptance. While the work of the production planner is greatly facilitated users that have to input parameters do not directly benefit from the system. Thus, the keys to the success of the system turned out to be the design of the data acquisition processes and especially the quality of the parameter input front-end. For the EPOS tool-parameter-sheets application special emphasis is put on the understandability, usability, and robustness of the system modules which are distributed to end-users. As a result, the controlled collaborative approach to maintain the simulation models has proven to be a solution to the problem of changing parameters. Still, management support remains a key factor during the introduction of new methodologies.

Due to the complexity, precedence constraints, and the importance of the concept for both applications — optimization strategies and integration of shop-floor-control systems — the system and its foundations have been developed by both authors. The summaries of the applications developed individually can be found in sections 10.5 and 11.5, respectively.

12.2. Outlook

The model of *integrated simulation* is inspired by questions from tactical production planning, see chapter 2, section 2.10, and figure 2.11. Whereas tactical questions are important for capacity and volume planners, there are numerous questions on the operational level which are especially important for manufacturing engineers and line control staff. Thus, an extension of the concept, the models involved, and the actual systems to support the

¹Pentium III, 800 MHz, 256MB RAM

decision-making process on the operational level are desirable. However, for further analyses on the operational planning level it is necessary to take more detailed information into consideration, like the state of machines (in process, maintenance, monitoring, etc.) and their set-up state, the distribution of work-in-process, priority disciplines employed, the availability of operator personal, etc. Further problems arise by a large ratio of lead time to the length of a planning period, i. e. if the lead time of parts through the production line is large compared to the time between fundamental changes in the production environment, the assumption of the existence of a steady state has to be carefully investigated. Moreover, short term management decisions like priorities for selected parts complicate the successful deployment of mathematical models for decision support purposes.

Nevertheless, there are various techniques and models available which can be integrated into the concept after successful extensions have been made. Optimal control strategies for deterministic systems are described in [MS95]. These have to be extended to optimal control of stochastic processes [G⁺86]. One approach are semi-Markovian decision processes for which stationary strategies can be calculated by the Howard algorithm [Nol81]. Other types of stationary strategies are covered in the field of optimal rate control policies for queueing systems [GH74].

The transient phase of queueing systems has been studied by different approaches. Linear approximations (fluid models) of multi-class queueing networks serve as a basis to derive scheduling strategies [CY93, CFY94, Wei95]. Diffusion approximation is a means to calculate the expected transient performance measures. The results by Mitzlaff [Mit97] have to be further extended to batch processing and queueing networks for successful use in production planning.

It has to be evaluated in how far results from scheduling theory [BESW93, Bru95] can be extended to handle real-time situations, uncertainty, and large production networks. Taking lot-sizing decisions into consideration leads to the problems of sequencing and lot-sizing [Tem95]. An annotated bibliography on lot-sizing is available in [Ke98].

Applying these techniques, the general task in operational production planning is to make deterministic models applicable in stochastic environments. The question for the application of any of the techniques presented is in how far parameters can be validly represented by some model to predict future values, whether they are deterministic, stochastic, stationary, or dynamic. Especially for stochastic approaches an important problem for

operational production planning remains: In stochastic environments short term observations might differ significantly from the expected long term averages. Moreover, any method mentioned needs further extensions to enable an integration with the product/work center/operation concept presented in section 3.

EPOS is currently not founded on a true object-oriented multi-tier architecture. Instead of an application server which controls and secures data integrity the planning data is often accessed directly on the database level. The reason lies in the ease-of-use of current database tools: It is easier to access a relational database via ODBC or JDBC than to access an application server by CORBA. Many commercially available tools for report creation have built-in support for ODBC and thus greatly facilitate the creation of reporting systems. The same is true for creating user front-ends.

The possibility of accessing the planning database directly is likely to be delimited for further systems to be connected: Planning results provided by EPOS need to be further integrated in enterprise resource planning (ERP) and supply chain management (SCM) systems. Instead of creating one dedicated interface per system to be integrated a connectivity approach based on middleware technologies like XML or the connector architecture of the J2EE 1.3 platform should be employed.

Some insufficiencies of the mathematical model concerning the representation of the real manufacturing processes have been lessened by modeling tricks or optimization methods. Research for a more general mathematical model is still needed, though. An approach to include an (a, b) batching rule in the analytical model is hindered by the open questions concerning the routing of batches through the network. The separate treatment of queues at process step level as opposed to work center level is currently investigated. This would allow the modeling of different batch sizes at a work center. Questions remain unanswered for networks with priority queues and operator availability on the cell level.

Even with more sophisticated processes to handle model parameters the trade-off between model complexity and effort needed for validation remains: A more detailed model demands more parameters and is thus harder to be validated. A possible enhancement of the validation process could be an additional module for automatic sensitivity analyses.

Currently, all production lines being simulated by EPOS are of a different type, i. e. there is no process that is performed in two different production lines. The system is able to handle this scenario, though. In addition to

the obvious advantages of sharing development and maintenance costs, a deployment of EPOS at a production line of the same type would expedite identity in the production planning processes, allow the comparison of different production lines regarding parameters and performance measures, and provide synergy effects.

Part IV

Appendices

Appendix

A

Database Tables of Sample Model

Throughout this work the structure of EPOS is explained by an accompanying example. The virtual **MeeKra inc.** company produces read/write heads and runs a wafer production line in Frankfurt. This appendix contains excerpts of the most important EPOS tables showing data from the **MeeKra inc.** production line in Frankfurt.

CELLNO	PRODLINENO	NAME	COMMENT
200	50	Lap	lapping and cleaning
210	50	Photo	coat, stepper, develop, inspect
230	50	Plate	plating
220	50	Sputter	sputter, sputter, ion mill
240	50	Test	measurements, inspections

Table A.1.: **Cell** table. The number of the production line (**ProdlineNo**) of the sample line is 50.

PGROUPNO	NAME	FORMNO	CURRENT	SIZENO
1030	Lambda	30	1	30
1040	Kappa	30	1	30
1050	Omega	30	0	30
1060	Sigma	30	1	30
1070	Taurus	30	1	30
1080	Olympia	30	1	30
1090	Mykonos	30	1	30
1100	Santorini	30	0	30

Table A.2.: ProductGroup table

PRODUCTNO	TYPENO	PGROUPNO	NAME
2620	30	1030	Lambda
2610	30	1040	Kappa
2530	30	1050	Omega
2660	30	1060	Sigma
2540	30	1070	Taurus
2640	30	1080	Olympia
2630	30	1090	Mykonos
2650	30	1100	Santorini

Table A.3.: Product table

SECTORNO	PROCESSNO	ID	NAME
2550	50	10	Start
2560	50	20	Layer
2570	50	30	Logistic

Table A.4.: Sector table. The process number of the sample line is 50.

OPERATIONNO	OPER	DESCRIPTION	SECTORNO
21310	100	Register wafer	2550
21320	110	clean surface	2560
21330	120	sputter under coat	2560
21340	130	lap surface	2560
42450	135	clean surface	2560
42430	140	apply resist 1	2560
21360	150	expose 1	2560
21370	160	develop 1	2560
21380	170	test photo resist 1	2560
21390	180	sputter layer 1	2560
21400	190	strip resist 1	2560
21410	200	apply resist 2	2560
21420	210	expose 2	2560
21430	220	develop 2	2560
21440	230	measure depth 2	2560
21450	240	visual test 2	2560
21460	250	sputter seed layer 2	2560
42440	255	plate NiFe layer 2	2560
21470	260	strip resist 2	2560
42460	270	apply 3	2560
42470	280	expose 3	2560
42480	290	develop 3	2560
42510	295	visual test	2560
42490	300	special beam 3	2560
42500	310	strip resist 3	2560
21480	500	sputter overcoat	2560
21490	510	final test	2560
21500	990	ship to customer	2570

Table A.5.: Operation table

ROUTINGTYPENO	NAME
0	Main flow
1	Rework
2	Rework flow
3	Back into main flow
4	Sector rework
5	Sector salvage

Table A.6.: RoutingType table

PGROUPNO	YEAR	WEEK	ROUTINGTYPENO	PRED	SUCC	ROUTINGPROB	SEQUENCE
1050	1990	1	0	21310	21320	1.0	500000
1050	1990	1	0	42430	21360	1.0	500005
1050	1990	1	0	42450	42430	1.0	500004
1050	1990	1	0	42460	42480	1.0	500011
1050	1990	1	0	42480	42510	1.0	500012
1050	1990	1	0	42490	42500	1.0	500014
1050	1990	1	0	42500	21480	1.0	500015
1050	1990	1	0	42510	42490	1.0	500013
1050	1990	1	0	21320	21330	1.0	500001
1050	1990	1	0	21330	21340	1.0	500002
1050	1990	1	0	21340	42450	1.0	500003
1050	1990	1	0	21360	21370	1.0	500006
1050	1990	1	0	21370	21380	1.0	500007
1050	1990	1	0	21380	21390	1.0	500008
1050	1990	1	0	21390	21400	1.0	500009
1050	1990	1	0	21400	42460	1.0	500010
1050	1990	1	0	21480	21490	1.0	500016
1050	1990	1	0	21490	21500	1.0	500017
1050	1990	1	1	21380	21340	0.022	
1050	1990	1	1	21440	21400	0.013	
1050	1990	1	1	21450	21400	0.129	
1050	1990	1	1	42510	21470	0.043	
1050	1990	1	1	21490	21340	0.051	

Table A.7.: Routing table. This table contains the flow for the product group *Omega* having the primary key 1050.

PGROUP NO	WORK CENTERNO	OPERATION NO	MEAN CT	SAMPLE	APPLY 1	APPLY 2	DEVELOP 1	DEVELOP 2	DEVELOP 3	BAKE CHILL	LONGEST STEP	CHILL TIME
1030	7640	21370	2.1	1.0	-1	0	0	0	0	4		
1030	7640	21410	2.4	1.0	-1	-1	0	0	0	12		
1030	7640	21430	2.6	1.0	0	0	-1	0	0	8		
1030	7640	42430	2.0	1.0	-1	0	0	0	0	4		
1070	7640	42430	2.0	1.0	-1	0	0	0	0	4		
1070	7640	21370	2.0	1.0	-1	0	0	0	0	8		
1070	7640	21410	2.0	1.0	-1	-1	0	0	0	10		
1070	7640	21430	2.0	1.0	0	0	0	-1	0	10		
1040	7640	42430	2.1	1.0	-1	0	0	0	0	10		
1040	7640	21370	2.1	1.0	-1	0	0	0	0	12		
1040	7640	21410	2.4	1.0	-1	-1	0	0	0	6		
1090	7640	21430	2.6	1.0	0	0	-1	-1	0	6		
1090	7640	42430	2.0	1.0	-1	0	0	0	0	6		
1090	7640	21370	2.0	1.0	-1	-1	0	0	0	6		
1090	7640	21410	2.0	1.0	-1	-1	0	-1	0	4		
1080	7640	21430	2.0	1.0	0	0	0	0	0			
1060	7650	21330	250.0	1.0	-1	0		0	0			
1060	7650	21390	200.0	1.0								
1060	7650	21480	500.0	1.0								
1070	7670	21330	63.0	1.0								
1080	7670	21390	63.0	1.0								
1050	7670	21330	63.0	1.0								
1050	7680	21360	5.0	1.0								
1100	7680	21360	5.0	1.0								
1060	7680	21360	5.0	1.0								
1030	7690	21360	5.0	1.0								
1060	7690	21360	5.0	1.0								
1070	7660	21390	1.0	1.0								10.0
1070	7660	21460	1.0	1.0								10.0
1070	7660	21480	2.0	1.0								14.0
1040	7870	21400	20.0	2.0							7.0	

Table A.8.: A sample of the ProcStep table. 32 of the 216 rows performed in the production line are shown.

WCTYPENO	NAME	PROCESSSTEPCLASS	WORKCENTERCLASS	COMPANYNO
50	Photo Cluster	sis.rw.extension.PhotoClusterProcessStep	sis.rw.extension.PhotoClusterWorkcenter	10
60	Sequence	sis.rw.extension.SequenceProcessStep	sis.rw.extension.SequenceWorkcenter	10
70	Inspection	sis.rw.extension.InspectionProcessStep	sis.rw.extension.ChillWorkcenter	10
80	Chill Required	sis.rw.extension.ChillProcessStep	sis.rw.extension.ChillWorkcenter	10

Table A.9.: WorkcenterType table

WORK-CENTERNO	NAME	CELL NO	WC-TYPENO	MTTR	MTBF	BREAKS	LUNCH	PFD	ENG'g-NEERING	MONT'-TORING	SETUP	LOADSIZE -MAX	LOADSIZE -MIN
7860	Brush Cleaner	200		4	100	0.75	1.50	0.75	2.0	10.0	2.0	12	12
7720	Lap tool	200		5	1300	0.75	1.50	0.75	3.0	0.0	15.0	10	5
7640	Photo Cluster	210	50	5	330	0.50	1.00	0.50	5.0	1.5	1.5	1	1
7680	Stepper 0815	210		5	200	1.00	1.00	0.50	1.0	10.0	5.0	1	1
7690	Stepper 4711	210		5	200	1.00	1.00	0.50	1.0	10.0	5.0	1	1
7710	NiFe Plate Cells	230	60	20	500	0.50	1.00	0.50	5.0	3.0	10.0	1	1
7870	Strip Tool 98x	230	60	5	1500	0.00	0.60	0.00	1.0	7.0	20.0	8	6
7650	Spratter R 2654	220	50	50	150	0.50	0.75	0.50	6.0	10.0	15.0	10	1
7660	Vacuum CV 4	220	80	50	300	0.75	1.50	0.75	3.0	4.0	14.0	10	1
7670	Ultra 5	220	80	100	200	1.50	1.00	5.00	10.0	10.0	20.0	1	1
7740	Microscope	240	70	1	1500	0.50	1.50	0.50	2.5	0.0	0.5	1	1
7750	Aut. Microscope	240	70	10	500	0.75	1.50	0.75	2.0	0.5	1.0	12	8
7760	Depth-Tester	240	70	3	150	0.50	1.00	0.50	8.0	3.0	0.5	1	1

Table A.10.: 16 of 52 columns are shown. For a list of a column see section 4.3.2

Appendix **B**

Abbreviations

ACL	Access Control List
API	Application Programming Interface
ASCA	Application Systems Control and Auditability
BOM	Bill-of-Materials
BU	Business Unit
CFM	Continuous Flow Manufacturing
COC	Center of Competence
CP	Process Capability Index
CPK	Critical Process Capability Index
CRM	Customer Relationship Management
CSV	Colon Separated Values (file format)
CORBA	Common Object Request Broker Architecture
DB	Database
DGR	Daily Going Rate
EA	Evolutionary Algorithm
EAI	Enterprise Application Integration
EJB	Enterprise Java Beans
ECS	Execution Control System
EPOS	Enterprise Planning and Optimization System
FIFO	First-In First-Out
FCFS	First-Come First-Serve

FLOCON	Floor Control System
FTY	First Time Yield
GA	Genetic Algorithm
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
HTML	Hyper Text Mark-up Language
IBM	International Business Machines
ISC	Integrated Supply Chain
ISO	International Standard Organisation
IT	Information Technology
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Standard Edition
JDBC	Java Database Connectivity
JDK	Java Development Kit
LAN	Local Area Network
ME	Manufacturing Engineering (Function)
MES	Manufacturing Execution System
MFG	Manufacturing
MPI	Manufacturing Process Instruction
MPS	Master Production Schedule
MRP	Materials Requirement Planning
MRP II	Material Resource Planning
MS	Microsoft
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
ODBC	Open Database Connectivity
OEM	Other Equipment Manufacturer
OLAP	On-line Analytical Processing
OS	Operating System
PC	Personal Computer
PCF	Property Control Facility
PCN	Program Change Notification
QA	Quality Assurance
RMI	Remote Method Invocation
SAN	Storage Area Network
SCM	Supply Chain Management
SFC	Shop-Floor-Control
SME	Simulation Modeling Environment

SPC	Statistical Process Control
SQL	Structured Query Language
SSD	Storage Systems Division
STD	Storage Technology Division
SW	Software
TPY	Throughput Yield
UML	Unified Modeling Language
PMC	Production Management Committee
WAN	Wide Area Network
WGR	Weekly Going Rate
WIP	Work-in-Process
WW	Worldwide
XML	Extended Markup Language

Interface of the Simulation Server

The following code shows the public interface of the simulation server in the interface definition language (IDL).

Module

```
#pragma prefix "IfM.PPS" // Institute for Mathematics,
                        // Production Planning Systems

module Ams {
    // forward declarations
    interface Device;
    interface Model;
    interface Workcenter;
    interface Operation;
    interface Product;
    interface Route;
    interface Need;

    struct ProfDataStruct {
        double rate;
        double lt;
        double eff;
    }
}
```

```

    double wip;
};

// some typedefs
typedef sequence<string>      StringSeq;
typedef sequence<Device>     DevSeq;
typedef sequence<Model>      ModelSeq;
typedef sequence<Workcenter> WcSeq;
typedef sequence<Operation>  OpSeq;
typedef sequence<Product>    PtSeq;
typedef sequence<Route>      RtSeq;
typedef sequence<Need>       NdSeq;
typedef sequence<ProfDataStruct> ProfSeq;

// Time constants
const long Time_mSec = 0;
const long Time_Sec  = 1;
const long Time_Min  = 2;
const long Time_Hour = 3;
const long Time_Day  = 4;
const long Time_Week = 5;
const long Time_Month = 6;
const long Time_Year = 7;

```

Exceptions

```

exception InternalError {string FileName;long LineNumber;};
exception DeviceNotFound {};
exception SyncFailed {};
exception RemoveFailed {};
exception Inconsistent {long Error;};
exception CalcError {};

```

Device

```

interface Device {
    readonly attribute string asset_num; // the devices ID (cannot be
    changed)

    Model    getModel();
    long     setName(in string Name);    // name, max. 12 characters
    string   getName();
    long     setDesc(in string Desc);    // description, max. 128 characters
    string   getDesc();
}

```

```

long    setXML(in string XML);
string  getXML();
long    setHidden(in boolean Hide);    // hide device from user in client
boolean getHidden();

boolean is_default(); // true if preset with default values

// force sync of device data to persistent storage
long sync(in boolean recurse) raises(SyncFailed);
long free();    // releases servant, thus frees up memory for
                // longer used devices and implicitly calls sync() before

// permanently removes device(s) from persistent storage:
long remove() raises(InternalError, RemoveFailed);

// lists asset numbers of devices to be deleted:
DevSeq list_remove();
};

```

Note that calling `free()` does not invalid any of the created references and variables. Calling `getName()` on a just freed model causes the server to create a new servant in this model (and maybe therefore swapping an old one out of memory) load the persistent data from storage and call the `getName()` method. So the following code is perfectly legal, but confusing:

```

Ams::Model_var mod = ctrl->load_model(modelsassetnum);
mod->free();
printf("The models name is %s\n",mod->getName());

```

Workcenter

```

interface Workcenter : Device {
    long set(in string name, in string desc, in long tools,
            in double rel, in double mdt, in double scvdt); // set all parameters
                                                         // in one method

    OpSeq listOp() raises(InternalError); // list all assigned op's
    long countOp(); // count all assigned op's
    long addOp(in Operation op) raises(InternalError); // add op to my
                                                         // internal Op-List

    long getTools(); // Number Of Tools
    long setTools(in long t);

    long setRel(in double Rel); // Reliability

```

```

double getRel();

long setMdt(in double MDT); // MeanDownTime
double getMdt();

// squared coefficient of variation the down time:
long setSCVDt(in double SCV);
double getSCVDt();

// calculated performance values
double WipQuantil(in double f);
double LeadTimeQuantil(in double f);
double QueueLengthQuantil(in double f);

double NoOfVisits();
double MeanServiceTime();
double SCVServiceTime();
double MeanComplTime();
double SCVComplTime();
double UtilizationNet();
double UtilizationTotal();
double MeanInputBatchSize();
double SCVInputBatchSize();
double MeanQueueLength();
double MeanWaitingTime();
double BatchedArrivalRate();
double SCVBatchedArrivalRate();
double LeadTime();
double Wip();
double Efficiency();
double LeadTimePt(in Product pt);
double WipPt(in Product pt);
double QueueLengthPt(in Product pt);
double NoOfVisitsPt(in Product pt);
double DensityFunctionWip(in long n);
double DistributionFunctionWip(in long n);
double DensityFunctionLeadTime(in double t);
double DistributionFunctionLeadTime(in double t);
};

```

Operation

```

interface Operation : Device {
    long set(in string name, in string desc,
            in Workcenter wc, in boolean source, in boolean sink,

```

```

// set all in one method:
in double mpt, in double scvpt, in long batch);

// the max. routing prob. not yet assigned:
readonly attribute double Scrap;
long    countRt();

// list all routings starting at this operation:
RtSeq listRt() raises(InternalError);
Route createRt(in Operation to, in double prob) raises(InternalError);

// returns the routing leading from this operation
// to the next operation:
Route    getRt(in Operation to) raises(InternalError);

// assigns this operation to a workcenter
long    setWc(in Workcenter wc) raises(InternalError);
Workcenter getWc() raises(InternalError);

Product getPt();
long    setPt(in Product pt) raises(InternalError);

// returns true for a sink operation (exactly one per product is needed)
boolean isSink();
long    setSink(in boolean Sink); // sets/removes the sink flag

// returns true if this operation gets input from an external source
boolean hasSource();
long    setSource(in boolean Source); // sets/removes the source flag

long    setMPT(in double mpt); // mean process time
double getMPT();

// squared coefficient of variation of process time
long    setSCVPT(in double scvpt);
double getSCVPT();

long    setBatch(in long batch); // batch size
long    getBatch();

// calculated performance values
double NoOfVisits();
double MeanQueueLength();
double LeadTime();
double GoodProb();
double RestLeadTimeGood();
};

```

Routing

```
interface Route : Device {
    long setFrom(in Operation op) raises(InternalError); //predecessor
    Operation getFrom();

    long setTo(in Operation op) raises(InternalError);    //successor
    Operation getTo();

    long setProb(in double prob);    // routing probability
    readonly attribute double prob;
};
```

Need

```
interface Need : Device {
    long setFrom(in Product pt) raises(InternalError);
    Product getFrom();

    long setTo(in Product pt) raises(InternalError);    // end product
    Product getTo();

    long setNeed(in double need);    // need for partial products
    readonly attribute double need;
};
```

Product

```
interface Product : Device {
    // set all parameters in one method
    long set(in string name, in string desc, in double demand);
    Operation createOp() raises(InternalError);
    OpSeq listOp() raises(InternalError);
    long countOp();
    Operation getSink();    // returns the sink operation
    OpSeq getSource();    // lists all operations
                        // connected to the products's source

    Need createNd(in Product To, in double need) raises(InternalError);
    NdSeq listNd();
    long countNd();

    long setDemand(in double demand);    // the demand in
                                        // good-parts-out per time unit
};
```

```

double getDemand();
long setTimeUnit(in long time);
long getTimeUnit();

long setInputBatch(in long InputBatch);
long getInputBatch();

long setSCVInputRate(in double SCV);
double getSCVInputRate();

long setContributionMargin(in double cm);
double getContributionMargin();

// calculated performance values
double OptPtMix();
double OptDemand();
double SecDemand();
double SecDemandYielded();
double PtMix();
double ArrivalRateMax();
double OverallLeadTime();
double Wip();
double Yield();
double RawProcessTime();
double Trigger();
double Efficiency();
double GoodPartsOut();
};

```

Model

```

interface Model : Device {
    // handle devices
    Workcenter createWc() raises(InternalError);
    Product createPt() raises(InternalError);
    long setTimeUnit(in long time);
    long getTimeUnit();
    long setHoursPerDay(in double d);
    double getHoursPerDay();
    long setDaysPerWeek(in double d);
    double getDaysPerWeek();
    long setDaysPerMonth(in double d);
    double getDaysPerMonth();
    long setDaysPerYear(in double d);
    double getDaysPerYear();
}

```

```

double TimeFactor(in long from, in long to);
long  setOptMaxUtilization(in double u);
double getOptMaxUtilization();
long  countWc();
long  countOp();
long  countPt();
long  countRt();
long  countNd();
WcSeq listWc() raises(InternalError);
OpSeq listOp() raises(InternalError);
PtSeq listPt() raises(InternalError);
RtSeq listRt() raises(InternalError);
NdSeq listNd() raises(InternalError);

string lastError();

// timeout values for automated database cleanups
// setting Lifetime = -1 sets the expire date (the only stored value)
// to 0 (= 00:00h(GMT)01.01.1970) which should be treated as infinite
// by the DB cleanup procedure
long setLifetime(in long days); // set lifetime in days (= 86400 seconds)
long getLifetime(); // remaining lifetime in days
long setExpire(in long secs); // set absolute timeout date in
                               // unix time_t format
long getExpire(); // when do we expire
                  //(Unix time_t format = seconds
                  // since 00:00h(GMT)01.01.1970)

// the calculation
readonly attribute boolean calc_running; // TRUE if calc() is running
long calc() raises(CalcError); // starts a calculation run
ProfSeq calcProfile(in long MaxIter);
long IsConsistent() raises(Inconsistent);
readonly attribute boolean has_changed;
void modify();

// calculated performance values
double Profit();
double OptProfit();
double Yield();
double LeadTime();
double Wip();
double ArrivalRate();
double ArrivalRateMax();
double GoodPartsOut();
double MaxThroughput();
double Utilization();

```

```
double UtilizationMax();
double UtilizationMin();
double NetUtilization();
double NetUtilizationMax();
double NetUtilizationMin();
double SCVUtilization();
double Trigger();
double RawProcessTime();
double Efficiency();
double LeadTimeGood();
};
```

Controller

The **Controller** object is the starting point to all queueing models. It can be used to create or load models.

```
interface Controller {
    Model create_model() raises(InternalError); // creates an empty model
    Model load_model(in string Name) raises(InternalError);
                                     // loads an existing model
    StringSeq list_names() raises(InternalError); // lists all known models
    void shutdown();
};
}; //end of module
```


Appendix

D

Legend for Work Center Parameters

Figure D.1 shows the work center **Assembly 1** with three operations assigned, **Ass1**, **Ass2**, and **Ass1rew**. The mean cycle time is given in the top right corner of the box for the operation, for example 2.5, 3.5, 4.5 time units for the three operations, respectively. The squared coefficients of variation for the operations are given in the bottom right corners of the operation boxes, 1, 1.5, and 1 in the example. In the bottom box of the work center

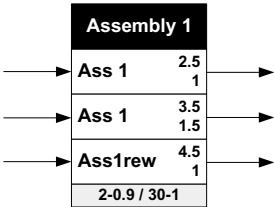


Figure D.1.: Sample work center

four parameters are given: the number of servers, the reliability, the mean down time, and the batch size at the work center. In the example these parameters are 2, 0.9, 30, 1, respectively. The squared coefficient of varia-

APPENDIX D. LEGEND FOR WORK CENTER PARAMETERS

tion of the mean down time of the work center is assumed to be 1.0 for all examples; this parameter is not shown explicitly in the figure.

Appendix E

The IBM Corporation

The International Business Machines Corporation located in Armonk, USA, is the holding company of the IBM Deutschland Speichersysteme GmbH. With a revenue of \$87.5 billion and 316,000 employees (see [IBM99a]) the IBM Corporation is one of the worldwide largest suppliers of information technology. Core activity is the production and distribution of hard and software. Supplementing the traditional domains, mostly newly founded service units are gaining more and more importance.

To react to complex requirements of different market segments, a comprehensive reorganization is taking place within the IBM (see figure E.1). The goal is to change the current decentralized organizational shape to a network of virtual teams. Different divisions have been founded which are organized according to the sectors' needs. Divisions within similar sectors are classified into groups. To better adapt to local market conditions geographic units supplement the product and sector-oriented divisions. In these units divisions are classified according to their geographic location. Thus, each group of the organization is simultaneously part of different virtual teams. The international solutions unit (ISU) overlays the divisions and units to ensure the advantage of international corporate practice. This organizational shape allows the IBM Corporation to align itself to national or sector-specific market situations while utilizing synergy and economy of size effects.

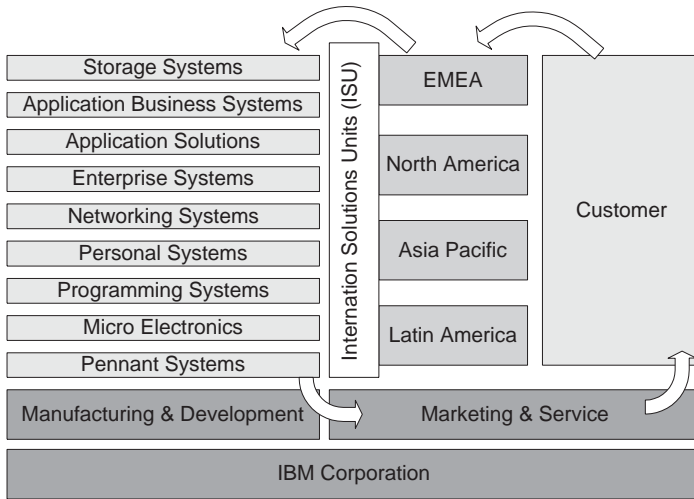


Figure E.1.: Organizational structure of the IBM Corporation

E.1. IBM Deutschland GmbH

The IBM Deutschland GmbH is one of the independent subsidiary corporations of the IBM Corporation which was founded in 1993 in Berlin. It replaced the preceding company, Deutsche Hollerith Maschinen Gesellschaft mbH that started in 1910 with the production of punch card machines. The 100% IBM subsidiary is a holding which combines numerous independent companies of the hardware, software, and services sectors. Being similar to its parent company, the IBM Deutschland GmbH shows a development from a manufacturing to a service-oriented enterprise. This can be observed by the changes in the proportion of revenue in each domain. Still, the hardware sector holds the largest proportion of the volume of trade which underlines its relative importance.

According to their business activity the companies within IBM Deutschland are mapped to the divisions of the IBM Corporation. The hardware oriented IBM Deutschland Speichersysteme GmbH is grouped along with the Storage Systems Division (SSD). The registered office of the IBM Deutschland Speichersysteme GmbH is Mainz where the manufacturing site is lo-



Figure E.2.: IBM plant Mainz

cated, as well. Together with the production lines in Hungary Mainz remains the last manufacturing site of the storage systems division in Europe.

E.2. The Manufacturing Plant in Mainz

The IBM manufacturing site for storage technologies is located on 34-hectare large premises, three kilometers from the center of Mainz, the state capital of Rhineland-Palatinate. In 1999 IBM employed more than 4,000 persons, about 3,000 of them in the IBM Deutschland Speichersysteme GmbH (see [IBM99b]). The main activity is the production of technology components for the use in hard disk drives. With the worldwide responsibility for IBM

storage systems and the guidance and logistic support of production sites in Hungary Mainz has the management competence for the IBM storage system division in Europe.

From the establishment of the site in 1965 to 1978 Mainz used to be mainly an assembly and test site. From the end of the seventies facilities have specialized on storage technology from hard disk drives to complete storage systems. Since 1995 the production in Mainz hosts the technologically critical processes for hard disk drives.

Discs and read/write heads from Mainz are deployed in IBM or OEM products. Above this, the IBM Speichersysteme GmbH develops storage systems software and offers complete storage solutions. Mainz has established itself as a development center for storage system software for the client/server market and for open systems. Within the Storage Systems Division (SSD) the IBM Deutschland Speichersysteme GmbH tightly cooperates with other production sites in Hungary, USA, Mexico, Japan, Thailand, and Singapore. Storage components, especially read/write heads, from Mainz are delivered to other producers of hard disk drives within the framework of OEM contracts.

The IBM site in Mainz also hosts the IBM Informationssysteme GmbH (ISG) with about 1,300 employees. It is their task to technically support, train, consult, and inform customers in whole Europe. Among others the ISG also contains the vocational training, a computing center, and the repurchase department.

Appendix F

Storage Systems Technology

Thinfil disk drives and magnetic heads from Mainz are assembled as complete disk drives and storage systems in other IBM plants or by business partners. Miniaturization of magnetic disk drives have improved capacity and performance and made it possible to create new applications using the optimized price/performance ratio. The worldwide market for storage systems and files is still growing very fast. Parallel to those changes the storage technology from large systems to laptops has merged during the past several years. Today the same magnetic disks and heads can be used in nearly every system environment. The relation between price and capacity of hard disk drives today is a thousand times better than ten years ago.

Figure F.1 shows the value chain of storage technology products. The individual processes can be classified into the production of technology components, assembly processes, and system integration.

Arrows in the charts identify the flow of material. One flow starts with wafers carrying thousands of read/write heads. These are cut into sliders which are mounted onto a so-called head gimbal assembly (HGA). Several HGAs are stacked to a head stack assembly (HSA) which is mounted on an actuator. In the HDD assembly line discs are stacked and assembled with the HSA. Together with the electronic controller card the hard disk is complete. Hard disk drives are distributed separately or are assembled to integrated storage systems.

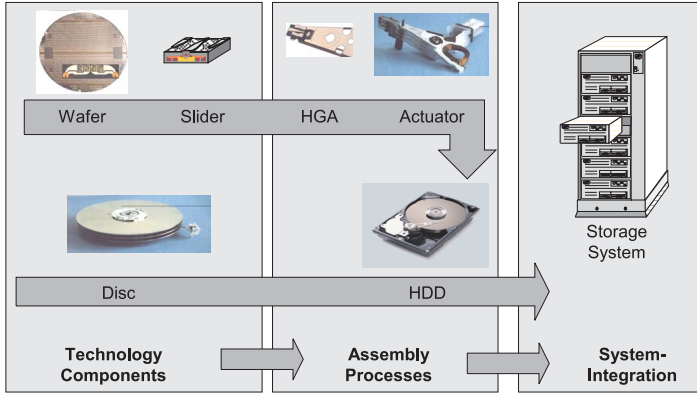


Figure F.1.: Storage technology value chain

In the following the important technological processes of the hard disk manufacturing are introduced. Nearly all operations need to be carried out in different categories of clean rooms.

F.1. Thinfilm Discs

A very important part of a drive is the thinfilm magnetic disk, a small disk made of aluminum or glass with a diameter of 2.5 or 3.5 inches. At the beginning of the manufacturing process the disks are polished and textured to gain defined surface conditions on both sides of the disk. Then the disk goes through a very complex vacuum sputtering process where a thin film of magnetic material is coated over a basis of nickel-phosphorus. This film is coated again with a thin protection layer of carbon. Before the first data can be stored on the disk, the magnetic film on both sides of the disk has to be formatted with thousands of concentric tracks. The distance between those tracks is around two micrometers. While reading or writing data the arm assembly and the whole actuator has to follow precisely one track with a very small tolerance.

F.2. The Wafer Process

In order to store more information on the disk, the area in which the data is stored gets smaller reducing the intensity of the magnetic field needed to code information. To read data stored in this way the sensitivity of the reading heads has to be increased. As the sensitivity of normal spools is no longer sufficient, today magnetoresistive heads are used which change their electrical resistance as a reaction to very small changes in the magnetic field of the disk. Those heads are manufactured on a wafer in a long and complex process. Currently up to 450 process steps are needed to structure the surface by building about 20 different layers using thinfilm techniques. Most steps in the wafer process can be summarized under:

- Sputtering isolation layers of aluminum-oxide
- Photo processes: coating, exposing, and developing photo resist
- Galvanic processes: nickel-iron layers, copper coil, gold studs
- Chemical etch processes
- Sputtering thin layers of metal onto the surface of the wafer
- Ion beam etching of metallic layers in vacuum
- Stripping of photo resist using organic detergents
- Cleaning by scrubbing and rinsing
- Measurements of layer thickness, widths, magnetism, and function
- Storage of wafers in nitrogen atmosphere

Several of these steps have to be repeated for successive layers many times on the same work centers. This leads to the re-entrant nature of semiconductor manufacturing lines.

F.3. The Slider Process

Each finished wafer contains some thousand read/write heads which have to be cut from the wafer. The process to cut the wafer into sliders is characterized by high micro-mechanical precision. First the wafers are cut into so-called rows which are polished in a process called lapping. Another thinfilm

process is needed to form the aerodynamic characteristics of the read/write heads by structuring the surface on which the heads fly above the disk. This process is similar to the wafer process and is once again technologically very challenging. From this time on the read/write heads are called sliders, they are still part of a row. Finally, the rows are cut to obtain the individual sliders with the magnetic element on the trailing edge.

The completed sliders are attached to suspensions and their magnetic elements are connected with cables thinner than a human hair. Magnetic heads and integrated amplifiers are assembled to a rotating positioner assembly, which has to position the sliders exactly over the tracks on the disk surface while reading or writing data.

Bibliography

- [A⁺91] David M. Amsden et al. *SPC simplified for services, practical tools, the continuous quality improvement*. Chapman & Hall, London, 1st edition, 1991.
- [AA90] Monte Aronson and Alvera L. Aronson. *SAS System: A Programmer's Guide*. McGraw-Hill, Inc., Cary, NC, 1990.
- [Ada97] Dietrich Adam. *Produktionsmanagement*. Gabler, Wiesbaden, 1997.
- [AFF00] J. Arnold, Th. Fischer, and A. Fröhner. Ein Modell zur simultanen Dimensionierung und Strukturierung von Fertigungssystemen mittels Genetischer Algorithmen. In J. Biethahn, editor, *7. Symposium: Simulation als betriebliche Entscheidungshilfe: Neuere Werkzeuge aus der Praxis*, pages 151–167, Braunlage, 2000.
- [Ahm98] Suhail M. Ahmed. *CORBA Programming*. Unleashed. SAMS, 1998.
- [Apa] Apache software foundation. <http://www.apache.org>.
- [Arn00] S. Arndt. Optimierung mit verteilten Simulationmodellen im Linux-Cluster. In Möller [Möl00], pages 105–110.

- [AvL85] E. H. L. Aarts and P. J. M. van Laarhoven. Statistical cooling: A general approach to combinatorial optimization problems. *Philips Journal of Research*, 40:193–226, 1985.
- [B⁺01a] E. J. W. Boers et al., editors. *Applications of Evolutionary Computing, EvoWorkshops 2001*, Como, 2001. Springer. Lecture Notes in Computer Science 2037.
- [B⁺01b] J. Bonal et al. A statistical approach to cycle time management. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 2001.
- [Bat94] Wolf-Dieter Batz. *Das SAS Survival-Handbuch*. Springer, Heidelberg, 1994.
- [Ber90] Klaus Bernecker. *SPC3, Anleitung zur Statistischen Prozesslenkung (SPC)*, volume 1 of *DGQ Schrift Nr. 16-33*. Beuth Verlag, Berlin, 1990.
- [BESW93] J. Blazewicz, K. Ecker, G. Schmidt, and J. Weglarz. *Scheduling in Computer and Manufacturing Systems*. Springer, Berlin, 1993.
- [Bey91] Ingward Bey, editor. *Simulation in CIM*. Springer, Berlin, 1991.
- [BGR94] Klaus Bichler, Wolfgang Gerster, and Rupert Reuter. *Logistik-Controlling mit Benchmarking*. Gabler, Wiesbaden, 1994.
- [Bha69] U. Narayan Bhat. Sixty years of queueing theory. *Management science*, 15(6):B-280–B-294, February 1969.
- [BmJ60] E. Brockmeyer, H. L. Halstrøm, and Arne Jensen. *The Life and Works of A. K. Erlang*. Applied Mathematics and Computing Machinery Series No. 6. Acta Polytechnica Scandinavica, second edition, 1960.
- [BN71] K. Bauknecht and W. Nef, editors. *Digitale Simulation*. Lecture Notes in Operations Research and Mathematical System. Springer, Berlin, 1971.

-
- [Bol83] G. Bolch. Approximation von Leistungsgrößen symmetrischer Mehrprozessorsysteme. *Computing*, 31:305–315, 1983.
- [Boo94] Grady Booch. *Objektorientierte Analyse und Design*. Addison-Wesley, Bonn, 1994.
- [BR72] U. Narayan Bhat and S. Subba Rao. A statistical technique for the control of traffic intensity in the queueing systems M/G/1 and GI/M/1. *Operations Research*, 20:955–966, 1972.
- [BR96] F. G. Boebel and O. Ruelle. Cycle time reduction program at ACL. *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 1996.
- [Bro95] Frederick P. Brooks. *The mythical man month*. Essays on software engineering. Addison-Wesley, 1995.
- [Bru95] Peter Brucker. *Scheduling algorithms*. Springer, 1995.
- [BT88] Gabriel R. Bitran and Devanath Tirupati. Multiproduct queueing networks with deterministic routing, decomposition approach and the notion of interference. *Management Science*, 34(1):75–100, January 1988.
- [BT89] Gabriel R. Bitran and D. Tirupati. Approximations for product departures from a single-server station with batch processing in multi-product queues. *Management Science*, 35(7):851–878, July 1989.
- [Bur56] Paul J. Burke. The output of a queueing systems. *Operations Research*, 4:699–704, June 1956.
- [Bur97] Cora Burger. *Groupware, Kooperationsunterstützung für verteilte Anwendungen*, volume 1. dpunkt.verlag, 1997.
- [Bus] Business objects. <http://www.businessobjects.com>.
- [Cam98] Camstar. *MESA Reference Manual*, volume 1. Camstar, June 1998.
- [CFY94] D. Connors, G. Feigin, and D. Yao. Scheduling semiconductor lines using a fluid network model. *IEEE Transactions on robotics and automation*, 10(2):88–98, April 1994.

- [CFY96] Daniel. P Connors, Gerald E. Feigin, and David D. Yao. A queuing network model for semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 9(3):412–427, August 1996.
- [Chu67] Kai Lai Chung. *Markov Chains with stationary transition probabilities*. Springer, Berlin, 1967.
- [CIM91] VDI-Gemeinschaftsausschuss CIM. *Produktionslogistik*, volume 5 of *Rechnerintegrierte Konstruktion und Produktion*. VDI Verlag, Düsseldorf, 1991.
- [CIM92] VDI-Gemeinschaftsausschuss CIM. *Qualitätssicherung*, volume 7 of *Rechnerintegrierte Konstruktion und Produktion*. VDI Verlag, Düsseldorf, 1992.
- [Cin75] E. Cinlar. *Introduction to Stochastic Processes*. Prentice-Hall, 1975.
- [CL81] Singha Chiamsiri and Micheal S. Leonard. A diffusion approximation for bulk queues. *Management Science*, 27(10):1188–1199, October 1981.
- [CM70] D. R. Cox and H. D. Miller. *The Theory of Stochastic Processes*. Methuen & Co, 1970.
- [CM94] H. Chen and A. Mandelbaum. Hierarchical modelling of stochastic networks. part I, II. In D.D. Yao, editor, *Stochastic Modeling and Analysis of Manufacturing Systems*, pages 47–105, 107–131. Spinger-Verlag, 1994.
- [Cob89] Cobuild. *English Learner’s Dictionary*. Collins Publishers, 1989.
- [Coh76] J. W. Cohen. *On Regenerative Processes in Queueing Theory*. Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, 1976.
- [Con63] R. W. Conway. Some tactical problems in digital simulation. *Management Science*, 10, 1963.

-
- [CR87] John M. Carroll and Mary Beth Rosson. Paradox of the active user. *Interfacing Thought, Cognitive Aspects of Human-Computer Interaction*, pages 80–111, 1987.
- [CRF97] Frank Chance, Jennifer Robinson, and John Fowler. Supporting manufacturing with simulation: Model design, development, and deployment. In J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, editors, *Winter Simulation Conference*, pages 1352–1356, 1997.
- [CRW99] F. Chance, J. Robinson, and N. Winter. Getting to good answers: Effective methods for validating complex models. *SMOMS Conference*, January 1999.
- [CS93] M. A. Centeneo and C. R. Standridge. Databases; Designing and developing integrated simulation modeling environments. In G. W. Evans, M. Mollaghasemi, E. C. Russel, and W. E. Biles, editors, *Proceedings of the 1999 Winter Simulation Conference*, page 526, Piscataway, NJ, 1993.
- [CSC93] R. Chandrasekharam, S. Subhranian, and S. Chaudhury. Genetic algorithm for the node partitioning problem and applications in VLSI design. *IEE*, 140(5):255–260, 1993.
- [Cur95] F. Curatelli. Implementation and evaluation of genetic algorithms for system partitioning. *International Journal of Electronics*, 78(3):435–447, 1995.
- [CY93] H. Chen and D. D. Yao. Dynamic scheduling of a multiclass fluid network. *Operations Research*, 41(6):1104–1115, 1993.
- [Dal67] D. J. Daley. Monte carlo estimation of the mean queue size in a stationary $GI/M/1$ queue. *Operations Research*, 16:1002–1005, December 1967.
- [Dan51] G. B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In Tj. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347, New York, 1951. Wiley.

- [Dan63] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963.
- [DC87] L. Davis and S. Coombs. Optimizing network link sizes with genetic algorithms. In M. Elzas, T. Oren, and B. P. Zeigler, editors, *Modelling and Simulation Methodology: Knowledge Systems Paradigms*, Amsterdam, 1987. North Holland.
- [DeJ75] K. A. DeJong. *An analysis of the behaviour of genetic adaptive systems*. PhD thesis, University of Michigan, 1975. Dissertation Abstr. Int. **36**, 5140B (Uni. Mikrofilm Nr. 76–9381).
- [Dia53] P. H. Diananda. Some probability limit theorems with statistical applications. *Proc. Cambridge Phil. Soc.*, 49:239–246, 1953. As cited in [Fis67].
- [DJS92] Julius Dufner, Uwe Jensen, and Erich Schumacher. *Statistik mit SAS*. Teubner Studienbücher, Stuttgart, 1992.
- [DL94] A. K. Dhingra and B. H. Lee. A genetic algorithm approach to single and multiobjective structural optimization with discrete–continuous variables. *International Journal for Numerical Methods in Engineering*, 37:4059–4080, 1994.
- [DPL94] S. Dauzere-Peres and J.-B. Lasserre. *An Integrated Approach in Production Planning and Scheduling*. Lecture Notes in Economics and Mathematical Systems. Springer, 1994.
- [DRL98] Jörg Domaschke, Jennifer Robinson, and Franz Leibl. Effective implementation of cycle time reduction strategies for semiconductor back-end manufacturing. In D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, editors, *Winter Simulation Conference*, pages 985–992, San Diego, December 1998.
- [DS90] P. Heymann D. and M. Sobel, editors. *Handbooks in OR and MS*, volume 2. Elsevier Science Publishers B.V., Elsevier, North Holland, 1990.
- [DS97] Markus Dierker and Martin Sander. *Lotus Notes 4.5 und Domino*. Addison-Wesley, Bonn, 1997.

-
- [DS98a] Lora Delwiche and Susan J. Slaughter. *The Little SAS Book*. SAS Institute, Cary, NC, second edition, 1998.
- [DS98b] Edgar Dietrich and Alfred Schulze. *Statistische Verfahren zur Qualifikation von Messmitteln, Maschinen und Prozessen*, volume 3. Carl Hanser Verlag, München, Wien, 1998.
- [Due93] G. Dueck. New Optimization Heuristics: The Great Deluge Algorithm and the Record to Travel. *Journal of Computational Physics*, 104(1):86–92, 1993.
- [DW99] J. Dai and G. Weiss. A fluid heuristic for minimizing makespan in job-shops. Preprint, to appear in *Operations Research*, <http://www.isye.gatech.edu/faculty/dai/reprints>, 1999.
- [Egl90] R. W. Eglese. Simulated annealing: A tool for operational reaserch. *European Journal of Operational Reasearch*, 46:271–281, 1990.
- [Eil69] Samuel Eilon. A simpler proof for $L = \lambda W$. *Operations Research*, 17:915–917, 1969.
- [F⁺97] Fahrmeier et al. *Statistik, Der Weg zur Datenanalyse*. Springer, Berlin, 1997.
- [FBM95] Michael Falk, Rainer Becker, and Frank Marohn. *Angewandte Statistik mit SAS*. Springer Lehrbuch, Berlin, 1995.
- [FF93] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Forrest [For93].
- [Fis67] George S. Fishman. Problems in the statistical analysis of simulation experiments: The comparison of means and the length of sample records. *Comm. ACM*, 10(2):94–99, February 1967.
- [Fis72] George S. Fishman. Estimation in multiserver queuing simulations. *Operations Research*, 22:72–78, 1972.
- [Fis73] George S. Fishman. Statistical analysis for queueing simulations. *Operations Research*, 20:363–369, 1973.

- [FK67] George S. Fishman and Philip J. Kiviat. The analysis of simulation-generated time series. *Management Science*, 13(7):525–557, 1967.
- [For93] S. Forrest, editor. *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, 1993.
- [FOW66] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley, Chichester, UK, 1966.
- [fQ90] Deutsche Gesellschaft für Qualität. *SPC 1*, volume 1. Beuth Verlag, Berlin, 1990.
- [Fra72] D. R. Frantz. *Non-linearities in genetic adaptive search*. PhD thesis, University of Michigan, 1972. Dissertation Abstr. Int. **33**, 5240B–5241B (Uni. Mikofilm Nr. 73–11, 116).
- [G⁺86] A. Göpfert et al. *Lexikon der Optimierung*. Akademie-Verlag Berlin, 1986.
- [Gel75] E. Gelenbe. On approximate computer system models. *Journal of the ACM*, 22:216–269, 1975.
- [GfQ79] Deutsche Gesellschaft für Qualität. *Begriffe und Formelzeichen im Bereich Qualitätssicherung*, volume 3. Beuth Verlag, Berlin, 1979.
- [GfQ80] Deutsche Gesellschaft für Qualität. *Stichprobenpläne für quantitative Merkmale (Variablenstichprobenpläne)*, volume 1. Beuth Verlag, Berlin, 1980.
- [GfQ81] Deutsche Gesellschaft für Qualität. *Stichprobenprüfung anhand qualitativer Merkmale*, volume 7. Beuth Verlag, Berlin, 1981.
- [GH74] D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 1974.
- [GJ79] M. Garey and D. Johnson. *Computers and Intractability*. W.H. Freeman, 1979.

-
- [Glo77] F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166, 1977.
- [Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, Mass., 1989.
- [Göp93a] Ingrid Göpfert. *Bedeutung und Gestaltung von Logistik-Kennzahlen für das Logistik-Controlling*, chapter 5.1, pages 223–232. In Weber [Web93], 1993.
- [Göp93b] Ingrid Göpfert. *Logistik Kennzahlen in der Praxis*, chapter 5.2, pages 232–251. In Weber [Web93], 1993.
- [Göt90] Thomas Götsche. *Einführung in das SAS-System für den PC*. Gustav Fischer, Stuttgart, 1990.
- [Got99] J. Gottlieb. *Evolutionary Algorithms for Constrained Optimization Problems*. Dissertation, Technische Universität Clausthal, Institut für Informatik, Shaker Verlag, Aachen, ISBN 3-8265-7783-3, 1999.
- [Got01] J. Gottlieb. On the feasibility problem of penalty-based evolutionary algorithms for knapsack problems. In Boers et al. [B⁺01a], pages 50–59. Lecture Notes in Computer Science 2037.
- [GP00] Björn Gehlsen and Bernd Page. Verteilte Ausführung von Simulationsstudien zur Optimierung komplexer Szenarien. In Möller [Möl00], pages 117–122.
- [Gro01] Object Management Group. OMG specifications and process. <http://www.omg.org/gettingstartedoverview.htm>, 2001.
- [GSS92] Johannes Gogolog, Rudolf Schümer, and Gerhar Ströhlein. *Datenverarbeitung und statistische Auswertung mit SAS*. Gustav Fischer, Stuttgart, 1992.
- [GT00] Hans-Otto Günther and Horst Tempelmeier. *Produktion und Logistik*. Springer, Berlin, 2000.
- [GTd93] F. Glover, E. Taillard, and D. de Werra. A user’s guide to tabu search. *Annals of Operations Research*, 41:3–28, 1993.

- [Gün93] Hans-Otto Günther. *Produktionsmanagement*. Springer, Berlin, 1993.
- [Haj00] Di. E. Hajrizi. Optimierungsabläufe von flexiblen Fertigungslinien durch Genetische Algorithmen. In Möller [Möl00], pages 357–262.
- [Han95] Thomas Hanschke. A queueing model for production planning. Technical report, IBM Speichersysteme GmbH, 1995.
- [Han98] Thomas Hanschke. Interner Bericht. Technical report, IBM Speichersysteme GmbH, 1998.
- [Har93] Joachim Hartung. *Statistik, Lehr- und Handbuch der angewandten Statistik*, volume 9. Oldenbourg, 1993.
- [Har96] J. Harrison. The bigstep approach to flow management in stochastic processing networks. in *Stochastic Networks, Theory and Applications*, edited by F. P. Kelly, S. Zachary and I. Ziedens, Clarendon Press, Oxford., 1996.
- [HC00] Ming-Der Hu and Shi-Chung Chang. Translating fab cycle time and output targets into productions control requirements for tool groups. IEEE Xplore, 2000.
- [Hei93] Claus E. Heinrich. *Logistik-Kennzahlen in der Standardsoftware der SAP AG*, chapter 5.4, pages 279–288. In Weber [Web93], 1993.
- [Hel94] S. Helber. *Kapazitätsorientierte Losgrößenplanung in PPS-Systemen*. Metzler & Poeschel Verlag für Wissenschaft und Forschung, Stuttgart, 1994. Dissertation.
- [HF99] M. Hickie and J. W. Fowler. Ancillary effects of simulation. In D. Sturrock P. Farrington, H. Nembhard and G. Evans, editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 754–758, Phoenix, December 1999.
- [Hol71] R. B. Hollstien. *Artificial genetic adaptation in computer control systems*. PhD thesis, University of Michigan, 1971. Dissertation Abstr. Int. **32**, 1510B (Uni. Mikrofilm Nr. 71–23, 773).

-
- [Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. 4. Druck, 1. MIT Press Auflage, 1992.
- [How68] R. A. Howard. The practicality gap. *Management Science*, 14(7):503–507, 1968.
- [HS90] Th. Hanschke and Th. Speck. AMS — An APL based modelling system for production planning. *IBM Systems Journal*, 1990.
- [HS00] Andreas Heuer and Gunter Saake. *Datenbanken: Konzepte und Sprachen*. MITP-Verlag, Bonn, 2000.
- [HU91] Péter Horváth and Georg Urban. *Qualitäts-Controlling*. C. E. Poeschel, Stuttgart, 1991.
- [HV99] Michi Henning and Steve Vinoski. *Advanced CORBA Programming with C++*. Addison-Wesley, Reading, Mass., 1999.
- [HW81] Philip Heidelberger and Peter D. Welch. A spectral method for confidence interval generation and run length control in simulations. *Communications of the ACM*, 24(4):233–245, April 1981.
- [HZ97a] Th. Hanschke and H. Zisgen. Fertigungsplanung mit XAMS. In A. Kuhn and S. Wenzel, editors, *Simulationstechnik, 11. Symposium in Dortmund*. Vieweg, 1997.
- [HZ97b] Th. Hanschke and H. Zisgen. XAMS - Ein neues Werkzeug zur Simulation von Fertigungslinien unter AIX-RS/6000. *Proceedings zum 6. Symposium, Simulation als betriebliche Entscheidungshilfe*, pages 75–89, 1997.
- [IBM] IBM Corporation, OS/2 Homepage. <http://www.ibm.com>.
- [IBM99a] IBM. Financial report 1999, international business machines and subsidiary companies. <http://www.ibm.com/investor/-financials/irfiar.phtml>, 1999.
- [IBM99b] Umwelterklärung 1999. IBM Report, Mainz, April 1999.

- [Inm99] W. H. Inmon. Data marts and the data warehouse: the information architecture for the millenium. Whitepaper: <http://www.billinmon.com/cif/datamart/kpdmstno.html>, 1999.
- [Inm00] W. H. Inmon. OLAP and Data Warehouse. Whitepaper: <http://www.billinmon.com/cif/datamart/kpdmstno.html>, 2000.
- [IS80] D.L. Iglehart and G. S. Shedler. *Regenerative Simulation of Response Times in Networks of Queues*, volume 26 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, 1980.
- [Jac57] James R. Jackson. Networks of waiting lines. *Operations Research*, pages 518–521, February 1957.
- [JBS97] Stefan Jablonski, Markus Böhme, and Wolfgang Schulze. *Workflow-Management*, volume 1. dpunkt.verlag, Heidelberg, 1997.
- [JC92] L. M. Jin and S. P. Chan. Analogue placement by formulation of macro-components and genetic partitioning. *International Journal of Electronics*, 73(1):157–173, 1992.
- [Jün89] R. Jünemann. *Materialfluß und Logistik*. Springer, 1989.
- [KB00] W. Krug and B. Baumbach. Parallele Optimierung für Statische und Dynamische Simulationsmodelle. In Möller [Möl00], pages 123–128.
- [KC00] W. Krug and G. Chmiel. Integration von Simulations- und Optimierungstools in der Produktionsplanung und Steuerung. In Möller [Möl00], pages 187–197.
- [Ke98] A. Kimms and A. Drexl (eds.). *Multi-Level Lot Sizing – An Annotated Bibliography*, pages 184–216. Springer, 1998.
- [KG96] Leonard Kleinrock and Richard Gail. *Queueing Systems, Problems and Solutions*. John Wiley & Sons, New York, 1996.

-
- [Kin64] J. F. C. Kingman. The heavy traffic approximation in the theory of queues. In *Proceedings of the Symposium of Congestion Theory*, Chapel Hill, 1964.
- [Kin72] J. F. C. Kingman. *Regenerative Phenomena*. John Wiley & Sons, London, 1972.
- [Kle75] Leonard Kleinrock. *Queueing Systems*, volume 1. Theory. Wiley Interscience, New York, 1975.
- [Kle76] Leonard Kleinrock. *Queueing Systems*, volume 2. Computer Applications. Wiley Interscience, New York, 1976.
- [Kle00a] Thomas Klein. Entwicklung eines Client-Server Systems zur integrierten Simulation. Master's thesis, Technische Universität Clausthal, Clausthal-Zellerfeld, 2000.
- [Kle00b] G. Klempert. *Solution of the Multi-Level Capacitated Lot-Sizing Problem by Simulated Annealing*. Dissertation, Technische Universität Clausthal, Institut für Mathematik, 2000.
- [Koz92] J. R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.
- [Kra98] M. Kramer. Ein integriertes Informationssystem unter dem Aspekt kooperativer Arbeit. Master's thesis, Technische Universität Clausthal, 1998.
- [KRW93] Axel Kuhn, Adolf Reinhard, and Hans-Peter Wiehndahl. *Handbuch Simulationsanwendungen in Produktion und Logistik*, volume 7 of *Fortschritte in der Simulationstechnik*. Vieweg, Wiesbaden, 1993.
- [KS90] Wolf-Michael Kähler and Werner Schulte. *SAS, Eine anwendungsorientierte Einführung*. Vieweg, Braunschweig, 1990.
- [KT75] Samuel Karlin and Howard M. Taylor. *A First course in stochastic processes*. Academic Press, New York, second edition, 1975.
- [KT81] Samuel Karlin and Howard M. Taylor. *A second course in stochastic processes*. Academic Press, New York, 1981.

- [KT97] Gerhard Keller and Thomas Teufel. *SAP R/3 prozessorientiert anwenden, Iteratives Prozess-Prototyping zur Bildung von Wertschöpfungsketten*, volume 2. Addison-Wesley, 1997.
- [KW00] W. Krug and T. Wiedemann. High-Performance-Optimierung mit ISSOP und SLX. In Möller [Möl00].
- [KWB92] A. Kuhn, Peter Wolf, and Gerhard Bandow. *Qualitätsmanagement im logistischen Informationssystem*, chapter 6, pages 141–184. Volume 3 of *Unternehmensführung und Logistik* [Pfo92], 1992.
- [Lav83a] Stephen S. Lavenberg. *Computer Performance Modeling Handbook*. Notes and Reports in Computer Science and Applied Mathematics. Academic Press, Orlando, 1983.
- [Lav83b] Stephen S. Lavenberg. *The Statistical Analysis of Simulation Results*, chapter 6, pages 267–329. In *Notes and Reports in Computer Science and Applied Mathematics* [Lav83a], 1983.
- [Law75] Averill M. Law. Efficient estimators for simulated queueing systems. *Management Science*, 22(1):30–41, September 1975.
- [Lea97] Robert C. Leachman. Closed-loop measurement of equipment efficiency and equipment capacity. *IEEE Transaction on Semiconductor Manufacturing*, 10(1), February 1997.
- [Leh90] John Lehoczky. *Statistical Methods*, chapter 6, pages 255–293. Volume 2 of D. and Sobel [DS90], 1990.
- [Lew60] R. W. Lewellyn. Control charts for queueing applications. *Journal of Industrial Engineering*, 11(4):332–335, July-August 1960.
- [LH97] Robert C. Leachman and David A. Hodges. Benchmarking semiconductor manufacturing. *Competitive Semiconductor Manufacturing Program*, 1997.
- [Lit61] J. D. C. Little. A proof for the queueing formulae $L = \lambda W$. *Operations Research*, 9:338–387, November 1961.

-
- [LK84] Averill M. Law and David Kelton. Confidence intervals for steady-state simulations: A survey of fixed sample size procedures. *Operations Research*, 32(6):1221–1239, November 1984.
- [LK91] A. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw Hill, Inc., New York, 2nd edition, 1991.
- [Mac61] Robert E. Machol. Standards on queue theory. *Operations Research*, pages 139–140, August 1961.
- [Mag99] C. Magalaras. Dynamic scheduling in multiclass queueing networks: Stability under discrete-review policies. *QUESTA*, 31:171–206, 1999.
- [Mag00] C. Magalaras. Discrete-review policies for scheduling stochastic networks: Trajectory tracking and fluid-scale asymptotic optimality. *Annals of Applied Probability*, 10(3), 2000.
- [Mar98] Donald P. Martin. The advantages of using short cycle time manufacturing (SCM) instead of continuous flow manufacturing (CFM). 1998 IEEE/SEMI Advanced Semiconductor Manufacturing Conference, 1998.
- [Mee97] Ingo Meents. Genetische Algorithmen für das Group-Technology Problem. Master’s thesis, Technische Universität Clausthal, 1997.
- [Mee01] I. Meents. A genetic algorithm for the group technology problem. In Boers et al. [B⁺01a], pages 90–99. Lecture Notes in Computer Science 2037.
- [MF00] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*, volume 2. Springer, Berlin, 2000.
- [Mic96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.
- [Mit97] U. Mitzlaff. *Diffusionsapproximation von Warteschlangensystemen*. PhD thesis, Technische Universität Clausthal, 1997.
- [Möl92] D. P. F. Möller. *Modellbildung, Simulation und Identifikation dynamischer Systeme*. Springer, 1992.

- [Möl00] D. P. F. Möller, editor. *Frontiers in Simulation – 14. Symposium Simulationstechnik*, Hamburg, 2000. SCS European Publishing House.
- [MP90] R. Mathar and D. Pfeifer. *Stochastik für Informatiker*. Teubner Stuttgart, 1990.
- [MPR95] A. K. Majhi, L. M. Patnaik, and S. Raman. A genetic algorithm-based circuit partitioner for MCMs. *Microprocessing and Microprogramming*, 41:83–96, 1995.
- [MS95] J. Macki and A. Strauss. *Introduction to optimal control theory*. Springer, 1995.
- [Mül91] Paul Heinz Müller, editor. *Lexikon der Stochastik*, volume 5 of *Wahrscheinlichkeitsrechnung und Mathematische Statistik*. Akademie Verlag, Berlin, 1991.
- [Nag92] Willi Nagl. *Statistische Datenanalys mit SAS*. Campus, Frankfurt/Main, 1992.
- [Nie01a] Jakob Nielsen. Novice vs. expert users. <http://www.useit.-com/20000206.html>, 2001.
- [Nie01b] Jakob Nielsen. Security and human factors. <http://www.useit.-com/20001126.html>, 2001.
- [Nol81] V. Nollau. *Semi-Markovsche Prozesse*. Verlag Harri Deutsch, 1981.
- [NS93] Bernd Noche and P. Scholtissek. *Anwendungen der Simulation in der Unternehmensplanung*, chapter 1.1, pages 7–42. Volume 7 of *Fortschritte in der Simulationstechnik* [KRW93], 1993.
- [NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.
- [OB00] A. Ostermann and U. Bracht. Neuronale Netze helfen mit: Hybride Materialflusssimulation zur Effizienzensteigerung im Umgang mit großen Simulationsmodellen. In Möller [Möl00].

-
- [OF86] John S. Oakland and Roy F. Followell. *Statistical Process Control*. Heinmann Newnes, Halley Court, Jourdan Hill, Oxford, 2 edition, 1986.
- [Oph96] Gregor Ophhey. Ein System zur Leistungsbemessung von Produktionsanlagen, Daten und Programmiermodell. Master's thesis, Technische Universität Clausthal, Clausthal, 1996.
- [OW99] Scott Oaks and Henry Wong. *Java Threads*. O'Reilly, 1999.
- [PA86] G. Pujolle and W. Ai. A solution for multiserver and multiclass open queueing networks. *INFOR*, 24:221–230, 1986.
- [Pag63] E. S. Page. On monte carlo methods in Congestion Problems: II. Simulation of Queueing Systems. *Operation Research*, 13:300–305, March 1963.
- [Pau96] Lutz Paulmann. Genetische Algorithmen für das Fixed-Charge-Transportproblem. Master's thesis, Institut für Informatik, Technische Universität Clausthal, 1996.
- [Pay88] James A. Payne. *Introduction to Simulation*, volume 1 of *Computer Science Series*. McGraw-Hill, Singapore, 1988.
- [Pfl96] Georg Ch. Pflug. *Optimization of Stochastic Models*. Kluwer Academic Publishers, 1996.
- [Pfo92] Chr. Pfohl. *Total Quality Management in der Logistik*, volume 3 of *Unternehmensführung und Logistik*. Erich Schmidt Verlag, Darmstadt, 1992.
- [PHP] PHP Development Team. <http://www.php.net>.
- [PS94] Bernd-Uwe Pagel and Hans-Werner Six. *Software Engineering*. Addison-Wesley, 1994.
- [PSU88] A. L. Peressini, F. E. Sullivan, and J.J. Uhl, Jr. *The Mathematics of Nonlinear Programming*. Springer, 1988.
- [PTVF88] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

- [Rec73] I. Rechenberg. *Evolutionstrategie*. Friedrich Frommann Verlag, Stuttgart, 1973.
- [Reh00] Jens Rehaag. EPOS Analyzer. Studienarbeit, Technische Universität Clausthal, 2000.
- [RFN00] J. Robinson, J. Fowler, and E. Neacy. Capacity loss factors in semiconductor manufacturing. Draft, January 2000. 15.
- [RGWAC99] Jennifer K. Robinson, Navdeep S. Grewal, Timbur M. Wulf, and Bruska Alvin C. Validating simulation model cycle times at Seagate Technology. In *Proceedings of the 1999 Winter Simulation Conference*, Phoenix, 1999.
- [RK73] M. Reiser and H. Kobayashi. On the accuracy of diffusion approximation for queuing systems. Technical report, IBM Research, Yorktown Heights, New York, September 1973.
- [RM91] Horst Rinne and Hans-Joachim Mittag. *Statische Methoden der Qualitätssicherung*. Hanser, München, 1991.
- [Röm01] Kai Römer. Mico. <http://www.mico.org>, 2001.
- [Ros85] Arlyn Custer Rosander. *Applications of Quality control in the service industries*. Quality and Reliability. Marcel Dekker, Inc., New York, 1985.
- [RS93] R. Y. Rubinstein and A. Shapiro. *Discrete Event Systems — Sensitivity Analysis and Stochastic Optimization by the Score Function Method*. John Wiley & Sons Ltd, 1993.
- [Rya89] Thomas P. Ryan. *Statistical Methods for Quality Improvement*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1989.
- [S⁺56] Bonnie B. Small et al. *Statistical Process Control*. Western Electric, Indianapolis, sixth edition, 1956.
- [Saa57] Thomas L. Saaty. Résumé of useful formulas in queueing theory. *Operations Research*, 5(2):161–200, April 1957.

-
- [Sac97] Lothar Sachs. *Angewandte Statistik*, volume 8. Springer, Berlin, 1997.
- [SAS99] *SAS/QC User's Guide*. SAS Institute Inc., Cary, North Carolina, version 8 edition, 1999.
- [Sch68] H. P. Schwefel. Experimentelle Optimierung einer Zweiphasendüse. Technical Report 35, AEG Forschungsinstitut, Berlin, 1968.
- [Sch75] H. P. Schwefel. *Evolutionstrategie und numerische Optimierung*. PhD thesis, Technische Universität Berlin, Mai 1975.
- [Sch77] H. P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels Evolutionsstrategie*. Birkhäuser Verlag, Basel, 1977.
- [Sch87] B. Schmidt. Modellaufbau und Validierung. In J. Biethahn and B. Schmidt, editors, *Fachberichte Simulation – Simulation als betriebliche Entscheidungshilfe*, pages 53–60. Springer, 1987.
- [Sch90] Bruce Schmeiser. *Simulation Experiments*, pages 295–330. Volume 2 of D. and Sobel [DS90], 1990.
- [Sch93] Rolf Schwinn. *Betriebswirtschaftslehre*. Oldenbourg, München, 1993.
- [Sch98] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & sons, 1998.
- [Sch99] Christoph Schneeweiss. *Einführung in die Produktionswirtschaft*. Springer Verlag, Berlin, Heidelberg, 1999.
- [SD95] N. Srinivas and K. Deb. Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.
- [SHF94] E. Schöneburg, F. Heinzmann, and S. Feddersen. *Genetische Algorithmen und Evolutionsstrategien*. Addison-Wesley, 1994.

- [SJ97] K. Srinivasan and Sundaresan Jayaraman. Integration of simulation with enterprise models. In S. Andradóttir, K.J. Healy, D. H. Withers, and B. L. Nelson, editors, *Winter Simulation Conference*, pages 1352–1356, New York, 1997.
- [Smi89] S. B. Smith. *Computer based production and inventory control*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1989.
- [Som96] Ian Sommerville. *Software Engineering*. Addison-Wesley, 1996.
- [Spe91] Eckehardt Spenhoff. *Prozessicherheit durch statistische Versuchsplanung in Forschung, Entwicklung und Produktion*. Gesellschaft für Management und Technologie, München, 1991.
- [Spu92] G. Spur, editor. *Datenbanken für CIM*. CIM-Fachmann. Springer, Berlin, 1992.
- [ST93] A. E. Smith and D. M. Tate. Genetic optimization using a penalty function. In Forrest [For93].
- [Ste90] W. J. Stevenson. *Production/Operations Management*. Richard D. Irwin, Inc., 3rd edition, 1990.
- [Str92] Bjarne Stroustrup. *Die C++ Programmiersprache*, volume 6. Addison-Wesley, Bonn, 1992.
- [Sun00a] Sun Microsystems Inc. Java 2 Platform, Standard Edition, v. 1.3 API Specification. <http://java.sun.com/j2se/1.3/docs/api/index.html>, 2000.
- [Sun00b] Sun Microsystems Inc. The Java Tutorial. <http://java.sun.com/docs/books/tutorial/index.html>, 2000.
- [Sun00c] Sun Microsystems Inc. The Source for Java Technology. <http://www.javasoft.com>, 2000.
- [Swa00] P. M. Swamidass, editor. *Encyclopedia of Production and Manufacturing Management*. Kluwer Academic Publishers, 2000.
- [SX93] M. Schönauer and S. Xanthakis. Constrained GA Optimization. In Forrest [For93].

-
- [Tam97] Randall A. Tamura. *Lotus Notes and Domino Server 4.5*. Unleashed. SAMS, 1997.
- [Tem95] H. Tempelmeier. *Material Logistik*. Springer, 1995.
- [Try94] W. J. Trybula. Developing simulation models without data. *1994 SMC Conference*, pages 201–204, 1994. San Antonio, Texas.
- [Tyb95] W. J. Tybula. Improving the modeling process. IEEE Explore, 1995.
- [Uhl82] Werner Uhlmann. *Statistische Qualitätskontrolle*, volume 2 of *Leitfäden der angewandten Mathematik und Mechanik*. Teubner, Stuttgart, 1982.
- [Ull88] Jeffrey D. Ullman. *Database and Knowledge-Base Systems*. Computer Science Press, 1988.
- [VDI93] VDI. Simulation von Logistik-, Materialfluß- und Produktionssystemen. VDI-Richtlinie 3633, Blatt 1, VDI-Verlag, Düsseldorf, 1993.
- [Vos00a] Werner Voss, editor. *Taschenbuch der Statistik*. Fachbuchverlag Leipzig, München, 2000.
- [Vos00b] Gottfried Vossen. *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*, volume 4. Oldenbourg, München, 2000.
- [WC90] Donal J. Wheeler and David S. Chambers. *Understanding Statistical Process Control*, volume 1. Addison-Wesley, 1990.
- [Web93] Jürgen Weber, editor. *Praxis des Logistik-Controlling*. Schriftenreihe der Wissenschaftlichen Hochschule für Unternehmensführung Koblenz. Schäffer-Poeschel Verlag, Stuttgart, 1993.
- [Web95] Jürgen Weber, editor. *Kennzahlen für die Logistik*. Schriftenreihe der Wissenschaftlichen Hochschule für Unternehmensführung Koblenz. Schäffer-Poeschel Verlag, Stuttgart, 1995.

- [Web98] Joseph L. Weber. *Using Java 2*. QUE, special edition, 1998.
- [Wei95] G. Weiss. On optimal draining of fluid re-entrant lines. In F.P. Kelly and R.J. Williams, editors, *Stochastic Networks*, Volume 71 of *IMA volumes in Mathematics and its Applications*, F. Kelly and R. Williams, editors. Springer, New York, pp. 93-105., 1995.
- [Whe91] D. J. Wheeler. Shewhart's chart: Myths, facts, and competitors. *45th Annual Quality Congress Transactions, American Society for Quality Control*, pages 553–538, 1991. As cited in [SAS99].
- [Whi83] W. Whitt. The queueing network analyser. *Bell System Technical Journal*, 62:2779–2825, 1983.
- [Win75] Robert L. Winkler. *Statistics, Probability, Inference, and Decision*, volume 2 of *Series in Quantitative Methods for Decision Making*. Holt, Rinehard and Winston, New York, 1975.
- [WJ99] Claus Weihs and Jutta Jessenberger. *Statistische Methoden zur Qualitätssicherung und -optimierung*. Wiley-VCH, Weinheim, 1999.
- [WM93] S. Wenzel and R. Meyer. *Kopplung der Simulation mit Methoden des Datenmanagements*, chapter 3.1, pages 347–368. Volume 7 of *Fortschritte in der Simulationstechnik* [KRW93], 1993.
- [Zäp82] Günther Zäpfel. *Produktionswirtschaft — Operatives Produktions-Management*. de Gruyter, 1982.
- [Zis99] H. Zisgen. *Warteschlangennetzwerke mit Gruppenbedienung*. PhD thesis, Technische Universität Clausthal, 1999.
- [Zit99] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Phd thesis, Graduate School of Engineering of the Air Force Institute of Technology, 1999.

-
- [ZRV00] G. Zülch, S. Rottinger, and T. Vollstedt. Entwicklung von Optimierungsstrategien bei der Simulation des mittel- und kurzfristigen Personleinsatzes. In Möller [Möl00], pages 305–312.

List of Figures

1.1.	Hierarchy of operations management decisions	8
1.2.	Typology of production systems	11
1.3.	Cycle time line profile	14
1.4.	Modeling and simulation	18
2.1.	Conventional and integrated simulation	30
2.2.	Model of integrated simulation	31
2.3.	Abstract concept of user roles	36
2.4.	Multi-tier architecture of integrated simulation	47
2.5.	Transformation of sequence work center	54
2.6.	Schematic view of the real-world photo cluster	56
2.7.	Sub-model for cluster develop stations	57
2.8.	Sub-model for cluster apply stations	58
2.9.	Model transformation steps of integrated simulation	63
2.10.	Process of capacity, lead time, and work-in-process planning	68
2.11.	Use of integrated simulation	70
3.1.	General notation of a queueing station	76
3.2.	Streams used in the network decomposition approach	89
4.1.	Overview of the EPOS architecture	100
4.2.	Company structure (core classes)	103
4.3.	Work centers	107

4.4.	Cycle time as a function of load size	111
4.5.	Work center types and groupings	112
4.6.	Plan work center	115
4.7.	Products	116
4.8.	Bill-of-materials	117
4.9.	Volume plans	119
4.10.	Routing	120
4.11.	Routing in the database	121
4.12.	Types of routing	122
4.13.	Staffing calendar	123
5.1.	Deployment of the tool-parameter-sheets	131
5.2.	Data input using a Java-applet	132
5.3.	Notes-view of the tool-parameter-sheets	138
5.4.	A tool-parameter-sheet as a Notes document	141
5.5.	Business process for the distributed parameter input	146
6.1.	Deployment of the simulator	149
6.2.	Static structure of the simulation server	152
6.3.	Example of a simple flow	155
6.4.	Inheritance, attributes, and performance measures	156
6.5.	EPOS Analyzer: Work flow and work center results	157
6.6.	EPOS Analyzer: Efficiency profile and bottleneck chart . . .	159
7.1.	Deployment of the automatic model generation	162
7.2.	The structure of the package SimSystem	163
7.3.	Interaction when a simulation run is started	164
7.4.	Requests for simulation runs	165
7.5.	Simulation runs	169
7.6.	Interaction diagram showing the loading of the real-world model	173
7.7.	Creation of objects on the simulation server	181
7.8.	Example of different types of routings between process steps	184
7.9.	Routing without modeling transportation	189
7.10.	Routing with a transportation operation at the place of a routing	190
7.11.	Mapping real-world objects to simulation server objects . .	192
7.12.	Mapping of a sequence work center.	193

7.13.	Work center requiring additional chill time after operations.	196
7.14.	Modeling inspections by routing batches around the inspection operation	198
7.15.	Schematic view of a photo cluster	199
7.16.	Mapping real-world and simulation objects of photo clusters	202
7.17.	Processes, production lines, and units	205
7.18.	Results of simulation run	211
8.1.	Deployment of the reporting subsystem	225
8.2.	Volume plan report	229
8.3.	Interactive report: Yield profile	232
8.4.	Overall results for production lines	234
8.5.	Work-in-process by cells	235
8.6.	Simulation results for work centers	236
8.7.	Results for products	238
8.8.	Capacity chart	239
8.9.	Capacity over time	240
8.10.	Results for products at work centers	241
8.11.	Lead time profile	242
8.12.	Events during model generation	243
8.13.	Tool-parameter-sheets and simulation results in Lotus Notes	246
9.1.	Application to administer the simulation parameters	250
9.2.	Screen-shot of the MS Access front-end	252
9.3.	The EPOS Administrator	254
9.4.	Simplified routing administration	258
10.1.	General scheme of an evolutionary algorithm	271
10.2.	Extension of interface definition of the simulation server . .	281
10.3.	EPOS result chart for product mix optimization	283
10.4.	EPOS Analyzer input panels for product mix optimization .	284
10.5.	EPOS Analyzer result panel for product mix optimization .	285
10.6.	Sample model for product mix optimization	286
10.7.	The simplex for the sample model	287
10.8.	Simple example of a junction	289
10.9.	Comparison of routing representations	291
10.10.	Insertion of artificial variables	293
10.11.	Algorithm to find junctions	296

10.12. Influence of ρ on work-in-process	297
10.13. Bottleneck hierarchy without optimized routings	300
10.14. Bottleneck hierarchy with optimized routings	300
10.15. Results of optimization experiments	303
10.16. Sample model	305
10.17. Performance measures of sample model	307
10.18. General crossover algorithm	316
10.19. $f(r, t)$ for $b = 2$ and $b = 5$	319
10.20. Goal function including penalty term for $r = 2$	325
10.21. Experiment 1: Traces (Operators 1–8)	330
10.22. Experiment 1: Traces (Operators 9 and 10)	331
10.23. Experiment 1: Overall results	332
10.24. Experiment 2: Traces	334
10.25. Experiment 2: Overall results	334
10.26. Experiment 3: Traces (Operators 1–8)	337
10.27. Experiment 3: Traces (Operators 9 and 10)	338
10.28. Experiment 3: Overall results (profit)	338
10.29. Experiment 3: Overall results (throughput goal)	340
10.30. Experiment 4: Traces (Operators 1–8)	341
10.31. Experiment 4: Traces (Operators 9 and 10)	342
10.32. Experiment 4: Overall results	343
11.1. Integration of simulation, SFC, and SAS	365
11.2. Process and waiting time at two different machines	367
11.3. Histogram process times	369
11.4. Quantile-Quantile Plots	370
11.5. Waiting time histogram	371
11.6. CV of inter-arrival time over the manufacturing process	372
11.7. Comparison between inter-arrival times at different processes	373
11.8. Inter-arrival time histogram	375
11.9. Relationship between work-in-process and lead time	377
11.10. Waiting time in front of a bottleneck	379
11.11. Correlograms of different waiting time processes	380
11.12. In and out-claim at successive process steps.	389
11.13. Quality control chart (process time)	397
11.14. Periodograms of the waiting time process	405
11.15. Quality control chart (waiting time)	415
11.16. Mapping shop-floor-control machines and work centers	420

11.17. Chart navigator	421
11.18. Effect of autocorrelation	423
11.19. Process and waiting time charts with non-matching targets	424
11.20. Estimation of the current utilization in the production line	427
11.21. Matching parts into the demanded weeks by using the expected date of outcome (<i>expectedDate</i>)	428
11.22. Expected output versus plan	430
11.23. Quantity of parts and average lateness	431
11.24. Mean queue length as a function of batch size.	435
11.25. Average batch size over production line.	438
 D.1. Sample work center	 473
 E.1. Organizational structure of the IBM Corporation	 476
E.2. IBM plant Mainz	477
 F.1. Storage technology value chain	 480

List of Tables

- 1.1. Hierarchy of capacity planning 12
- 1.2. Possibilities of adjusting capacity 16

- 2.1. Amount of time spent on particular simulation tasks 33

- 3.1. Structure of the model: Mappings of indices 79
- 3.2. Naming convention of model artifacts used in this section . . 79

- 4.1. Sample process steps for *Stepper 4711* 106
- 4.2. Attributes of class **Workcenter** for administration 108
- 4.3. Attributes of class **Workcenter** for reliability calculation . . 109
- 4.4. Attributes of class **Workcenter** for cycle time calculation . . 110
- 4.5. Attributes of class **Parameter** 113
- 4.6. Staffing attributes 124

- 5.1. Icons for work center state 140

- 6.1. Attributes of the classes of the simulation server 153

- 7.1. Attributes of class **SimRequest** 168
- 7.2. Attributes of class **Simulation** 170
- 7.3. Mapping real-world to simulation server classes 179
- 7.4. Differences between BOM and units 207

7.5.	Events generated during model generation.	221
9.1.	Evaluation of SQL commands	251
9.2.	Evaluation of the MS Access front-end	253
9.3.	Advantages and disadvantages of the EPOS Administrator	256
10.1.	Parameters for sample model	301
10.2.	Decision variables	310
10.3.	Experiment 1: Descriptive statistics	331
10.4.	Experiment 1: ANOVA	331
10.5.	Experiment 1: Post hoc analysis (Scheffé)	332
10.6.	Experiment 2: Descriptive statistics	333
10.7.	Experiment 2: ANOVA	335
10.8.	Experiment 2: Post hoc analysis (Scheffé)	335
10.9.	Experiment 3: Descriptive statistics	336
10.10.	Experiment 3: ANOVA	339
10.11.	Experiment 3: Post hoc analysis (Scheffé)	339
10.12.	Experiment 4: Descriptive statistics	340
10.13.	Experiment 4: ANOVA	342
10.14.	Experiment 4: Post hoc analysis (Scheffé)	343
11.1.	Structure of the transaction table	366
11.2.	Comparison between 2 estimation techniques	383
11.3.	Required variables for Shewhart control charts	393
11.4.	Experimental results	410
11.5.	Comparison of different batch size scenarios	439
A.1.	Cell table	453
A.2.	ProductGroup table	454
A.3.	Product table	454
A.4.	Sector table	454
A.5.	Operation table	455
A.6.	RoutingType table	456
A.7.	Routing table	456
A.8.	ProcStep table	457
A.9.	WorkcenterType table	458
A.10.	Workcenter table	458

Curriculum Vitae Martin Kramer

Birthday	October 17th 1971
Place of birth	Mülheim a.d. Ruhr
Nationality	German
Status	Single
Address	Willy-Brandt-Allee 13, 65197 Wiesbaden

Education

1978-1982	Primary school
1982-1991	Otto-Pankok-Schule, Gymnasium
1988-1989	Foreign exchange student at the Frank W. Cox High-School, Virginia Beach, VA, USA
1991	High school degree (Abitur)
1992	Study of computer science at the Technical University of Clausthal
Aug. 1994	Pre-diploma in computer science
Feb. 1995	Admission to the <i>Studienstiftung des Deutschen Volkes</i>
Sept. 1998	Diploma in computer science

Oct. 1998	IBM Scholarship for the preparation of the PhD-thesis, IBM Deutschland Speichersysteme GmbH, Mainz
14.12.2001	Examination for the doctorate

Military Service

10/1991 – 09/1992	Basic military service for German Federal Armed Forces (Bundeswehr), Lipperland-Kaserne, Lippstadt
-------------------	--

Foreign Languages

Business English (fluent)

French (4 years in school)

Portuguese (basics)

Practicals and Projects

03/1993 – 04/1993	Stahlwerk Ergste: Software and hardware installations
06/1994 – 08/1998	Stadtwerke Clausthal-Zellerfeld GmbH <ul style="list-style-type: none"> • McPower (multi-media game) • System for the integrated cash and profit planning • Internet site of the company • General IT support and consulting
03/1995 – 04/1995	Institut Straumann, Waldenburg, Switzerland: Design and implementation of a backup and retrieval strategy for an image processing system

08/1996 – 11/1996	IAESTE-scholarship for a practical at Curso Pré Universitário, Porto Alegre, RS, Brasil
03/1996 – 11/1997	Technical University of Clausthal: <ul style="list-style-type: none"> • Tutorials for the courses of <i>applied computer science</i> • Design of the home page of the multi-media project group
01/1997 – 06/1997	Marleaux Bass-Guitars: Design of a book-keeping system on basis of MS Access
07/1997 – 08/1998	IBM Speichersysteme GmbH, Mainz: Implementation of a workgroup system and an intranet site on basis of Lotus Notes and MS Access
08/1997 – 04/1998	Harz Treff Hotel, Hahnenklee: Implementation of a bookkeeping system

Studies

Seminars	Seminars during the studies <ul style="list-style-type: none"> • International seminar, Lessach, Austria; presentation in English: <i>Parallel Search</i> • Theoretical seminar: <i>Game Theory</i>
Academy	Summer academy of the <i>Studienstiftung des Deutschen Volkes</i> , participation in the group <i>Adaptive Computer and Program Structures</i>
Software-Practical	Design and implementation of the application <i>Virtual Keys</i> , a system for real-time transformation of MIDI data

Master Thesis

An integrated information system under the aspect of collaborative work (in Cooperation with IBM Deutschland Speichersysteme GmbH, Mainz)

Hobbies

Music

Piano and electronic instruments (jazz, funk, fusion)

Sport

Fitness, badminton, skiing, sailing

Web Design

Design and implementation of web sites

- www.marleaux-bass.de
Producer of electric basses
- www.brigitte-weber.de
Management consulting for internal and external corporate communications, management training
- www.druckereiservice.de
reico GmbH, printing service
- www.stadtwerke.clausthal.harz.de
- www.in.tu-clausthal/~mmedia/
Multi-media project group of the TU Clausthal
- www.materials-consult.de
Technology consulting, special materials, medicine technology, and aerospace

Curriculum Vitae Ingo Meents

Date of Birth	March 21 st 1971
Place of Birth	Wilhelmshaven, Germany
Nationality	German
Marital status	Single
Address	Vogelsbergstraße 2, 55129 Mainz

Education

8/1983 – 5/1990	Käthe-Kollwitz High School, Wilhelmshaven
7.5.1990	High School Degree (Abitur)
10/1990 – 9/1997	Studies of Computer Science and Mathematics at the Technical University of Clausthal
24.7.1992	First Diploma in Computer Science (Vordiplom)
2.12.1992	First Diploma in Mathematics (Vordiplom)
15.9.1997	Final Diploma in Computer Science (Hauptdiplom)
10/1993 – 7/1994	Studies of Computer Science & Artificial Intelligence at the University of Edinburgh, Scotland
10/1998 – 2/2002	Scholarship by the IBM Deutschland Speichersysteme GmbH to achieve a PhD
14.12.2001	Examination for my doctorate

Civil Service

10/1997 – 9/1998 Medical School Hannover, Clinic for Cardiovascular and Thoracic Surgery

Work Experience

Scientific student job (Dept. of Computer Science, TU Clausthal)

10/1992 – 5/1993 Collaboration on the development of the rule-based programming language INTRAN

10/1994 – 2/1995 Teaching a programming course (language SCHEME)

10/1996 – 2/1997 Supervision of exercises in the area of neural networks

Scientific student job (Dept. of Mechanical Engineering, TU Clausthal)

10/1994 – 7/1995 Implementation and development of Artificial Intelligence tools in the area of CAD

Student job at the IBM Informationssysteme GmbH Hamburg

7/1994 – 10/1994 Collaboration on a large project (development of a configuration and change management tool) for the police and Ministry of Interior of Mecklenburg-Vorpommern, a federal state of Germany, in Schwerin: programming, documentation, requirements definition, insights into project management

Further experience

10/1996 – 2/1997 Translation of the manual and program dialogs of PRISMA into English and its presentation at the Fair GlasTec in Düsseldorf in 1996. PRISMA is a software product for the glass industry by the company *ilis Gesellschaft für integrierte Laborinformationssysteme*.

Publications (not including those based on this thesis)

Reviewed paper *A Genetic Algorithm for the Group-Technology Problem*, Applications of evolutionary computing, Workshop of Evolutionary Computation in Combinatorial Optimization (Proceedings), Como, April 2001, Springer Lecture Notes in Computer Science, Vol. 2037, pp. 90-99

Stays Abroad

8/1988, 3/1989 Dunfermline, Schottland, school exchanges
10/1993 – 7/1994 University of Edinburgh, Scotland, scholarship by the German Academic Exchange Service DAAD (Deutscher Akademischer Austauschdienst)

Foreign Languages

English and Business English fluent:

- 9 years at school (Leistungskurs)
- Several stays abroad in Scotland
- Adult evening classes for Business English: Cambridge Certificate in English for International Business and Trade

French, 5 years at school up to the 11th form

Hobbies

Music Piano, guitar (classics, jazz, funk, rock, big-band)
Sports Ballroom dancing (Standard/Latin), running, surfing, fitness training