LUND UNIVERSITY

On Discrete-Event Simulation and Integration in the Manufacturing System Development Process

Randell, Lars

2002

Link to publication

Citation for published version (APA):
Randell, L. (2002). *On Discrete-Event Simulation and Integration in the Manufacturing System Development Process*. [Doctoral Thesis (monograph), Department of Electrical and Information Technology]. Division of Robotics, Lund University, P.O.Box 118, 221 00 Lund, Sweden,.

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

On Discrete-Event Simulation and Integration in the Manufacturing System Development Process

Lars Randell

Division of Robotics
Department of Mechanical Engineering
Lund University , 2002

PhD Thesis
CODEN: LUTMDN/(TMMV-1054)/1-165/2002
ISBN 91-628-5319-8

On Discrete-Event Simulation and Integration in the Manufacturing System Development Process

Lars Randell

LUND UNIVERSITY

Division of Robotics
Department of Mechanical Engineering
Lund University, P.O. Box 118, SE-221 00 Lund, Sweden

PhD Thesis
CODEN: LUTMDN/(TMMV-1054)/1-165/2002
ISBN 91-628-5319-8

©2002 by Lars Randell and the Department of Mechanical Engineering, Lund University.
All rights reserved.

Printed in Sweden
KFS i Lund AB, Lund

It's the job that's never started as takes longest to finish, as my old gaffer used to say.

Samwise Gamgee, son of Hamfast
in The Fellowship of the Ring — The Lord of the Rings
by J.R.R. Tolkien

Summary

Comparing the usage of DES, CAD and RS in the industry shows that, although DES has been available for mainstream use for a long time, the usage differs significantly. Possible reasons for this situation are:

- lack of simulation knowledge,
- misuse of simulation,
- cost versus conceived benefits of using DES, and
- using DES is considered complex.

Lack of knowledge is an issue for educational institutions and can not be addressed within a thesis. Instead the focus has been on the other issues.

DES is seldom used in the manufacturing system development process, instead it is usually used to cure problems in existent systems, *fire fighting*. This has the effect that the simulation study alone is considered being the cost driver for the analysis of the manufacturing system. It is argued that this is not a entirely correct view since the analysis has to be performed anyway, and the cost directly related to the simulation study is mainly in the model realization. The fire fighting approach also has the effect that the simulation model is seldom reused, which in turn reduce efficiency.

There is a large base in the DES literature concerning technical issues. However, little is found concerning the management of the DES process and the infrastructure required to provide repeatable high quality simulation studies. A model, well established in the software domain, is therefore supplied to be used for management and engineering process improvements and for improvements of the organizational issues to support simulation activities. By institutionalizing the process via polices, standards, and organizational structures, an organization gains in process maturity. Institutionalization entails building an infrastructure and a culture that supports the methods, practices, and procedures so that the organization can continue to generate high quality simulation studies. By institutionalizing and utilizing well defined processes the conceived complexity related to DES is considered to be reduced over time.

It is concluded that it is preferred if the simulation study life cycle coincides with the corresponding manufacturing system's life cycle to increase the

usability of the simulation model and to increase efficiency in the simulation study process.

Cost is highly correlated to time. The presented methodology tries to reduce time consumption and lead-time in the simulation study by:

- reducing redundant work,
- reducing rework, and
- moving labor intensive activities forward in time.

A framework is provided to reduce the time to collect and analyze input data. The framework aims at delivering high granularity input data without dependencies. The input data collection framework is designed to provide data for operation and analysis of a manufacturing.

To reduce the model realization time two approaches are presented. The first approach supplies a set of modules that enables parameterized models of automated subassembly systems. The second approach builds and runs the simulation model based on a copy of an MRP database, i.e. there is no manual intervention required to build the simulation model. The approach is designed to forecast the performance of an entire enterprise. Since the model is generated from a database, the approach is highly scalable. Furthermore, the maintenance of the simulation model is reduced considerably.

Preface

The thesis is organized in parts with the contents described below.

Prologue

Chapter 1 presents the context in which this research has been performed. The research problem and objectives that constitutes the main threads for the thesis is presented. In Chapter 2 the chosen research method is discussed.

Frame of Reference

Chapter 3 briefly presents discrete-event simulation. Different facets of integration is presented in Chapter 4. The software configuration management methodology have been a major part of the research and is discussed and briefly compared to product data management in Chapter 5.

Case Studies

Four of the case studies performed are presented in Chapters 6-9.

Contributions

The presentations of the case studies are somewhat scattered and are tied together in Part IV. The base methodology is first presented and then a set of methods and tools that can be added in different contexts are presented in the following chapters.

Epilogue

Finally the thesis is discussed and concluded in Chapters 15 and 16. Chapter 17 presents possible future research.

Appendix

A few appendixes are supplied to supplement some of the chapters. Appendix A presents the proposed discrete-event simulation process with IDEFØ notation. Appendix B presents definitions of availability measures related to Chapter 12. Finally, Appendix C gives an example of the incremental generation of documents in the simulation study as presented in Chapters 10 and 11.

Lists and Indexes

In the end of the thesis there are lists of figures, tables, acronyms and a glossary.

Acknowledgments

Numerous of people and organizations has contributed to this research. First of all I would like to thank professor Gunnar Bolmsjö for his contributions to the research and the guidance throughout the work. My other colleagues at Division of Robotics, especially Lars Holst, have been great supporters and discussion partners.

The work has mainly been funded by the NUTEK, IT Verkstad, VSOP för CONSENSUS project.

The first project was made at Profilgruppen in Åseda, Sweden. The project team at Profilgruppen has truly set a new standard for short response times. Especially Stefan Korzonek and Charlotte Ehn were most helpful during the project.

The second case was performed at Volvo Car Body Components in Olofström in collaboration with ABB Body-in-White. At ABB Body-in-White I would like to thank Erling Samuelsson and Per-Ola Andersson who helped me to start and perform the project. At Volvo Car Body Components I would like to thank Peter Arvidsson, Camilla Arvidsson, Ferenc Mora, Anders Skogsgård, Jens Bergdahl and all those at the computer department that helped me. There have been a lot of people involved besides those mentioned here that has contributed in one way or another. Bengt-Göran Olsson at Volvo Car Corporation played an important role at the end of the project as he helped to structure the information and knowledge gained.

The third case performed at BT Products triggered the use of configuration management. Oliver Pallman, Jörgen Norgren at BT Products and Lars Holst contributed in this project. In the second case study at BT Products Hoa Hai Gieng, Michael Nilsson and Jörgen Norgren from BT Products and Lars Holst and Arne Ingemansson at Division of Robotics contributed.

At Volvo Articulated Haulers two Bachelor Thesis projects have been supervised. The kind of positive thinking and open mind that is in this company is rare and is probably one of the reasons of the company's success. As usual there are a lot of people that have contributed and I would especially like to mention Jan-Anders Rålin, Gunnar Genander and Tomas Lagnander. Lately another project at Volvo Articulated Haulers has been performed in collaboration with Martina Gustavsson and Tobias Johansson at Teknikcentrum Kronoberg.

Another Bachelor Thesis supervised was performed at Urshults Werkstads

AB. This interesting project was made possible by Ulf Hansén's and Kristian Petersson's initiatives and collaboration.

Hans-Göran Karlsson and Jonas Klein at Teknikcentrum Kronoberg have in various ways contributed to my work performed at Växjö University. Hans-Göran Karlsson has made a number projects possible with his wide network of contacts. Jonas Klein has contributed as a discussion partner and has given a lot to think about twice.

I have used QUEST for the simulation and have received a lot of support from Delfoi. Among those at Delfoi in Sweden that has helped me are Torbjörn Danielsson, Anders Danielsson and Jeanette Andersson. In Finland Tomi Kosunen and Ahti Rossi have helped me with tricky QUEST related problems.

It has been a pleasure to work with the people at Institutet för Verkstadsteknisk Forskning. Especially I would like to thank Bertil Gustafsson, Henrik Widlund and Hovik Moosakhanian.

There has been a lot of statistics in the work and Björn Holmqvist at the Department of Mathematical Statistics at Lund University has guided me through the mathematical jungle.

I would like to thank all members of my family that has supported me all the way.

Finally a special thanks to all those in Japan who made the stay such a pleasant experience. Our supervisor professor Hiroshi Fujii at Department of Mechanical and Systems Engineering at Gifu University, among many things, arranged industrial visits and guided us during our visit in Japan. Mr Itaru Tomei and Mr Hiroshi Mearashi took care of us during our stay in Gifu in the best way possible. The people of Japan are most friendly and always take care of their guests. Fujii-sensei, Tomei-san, and Mearashi-san with families presented the state-of-the-art hospitality common in Japan. Professor Toshihiko Mori at the Department of Mechanical Engineering at Nagoya University arranged several interesting industrial visits and guided us. Professor Etsuo Yokoyama at the Department of Education at Nagoya University helped us in many ways and gave us invaluable insights into the ways of Japan. The ones mentioned and several others gave us an invaluable insight into Japanese culture, experiences, and pleasant memories that I will keep for the rest of my life. It is difficult to express in words all the small details that made the visit such a success. Anyone who has tried to buy a train ticket from one of the machines at a Japanese station knows what I mean.

どうも有難う御座いました

Contents

Summary v

Preface vii

Acknowledgments ix

I Prologue 1

1 Context 3
1.1 Computer-Aided Manufacturing System Engineering 3
1.2 Integrated Concurrent Engineering 4
1.3 Research Problem 8
1.4 Research Objective 8

2 Materials and Methods 11
2.1 Scientific Method 12
2.2 Research Type 12
2.2.1 Objectivity and Subjectivity 13
2.2.2 Participant-Observation 13
2.2.3 Interviews 13
2.3 Case Studies 14
2.3.1 Single or Multiple Case Study Design 14
2.3.2 Group to Research on 14
2.4 Criteria for Judging the Quality of Research Designs 15
2.4.1 Construct Validity 15
2.4.2 Internal Validity 16
2.4.3 External Validity 16
2.4.4 Reliability 16
2.5 Discrete-Event Simulation Software 16
2.6 Configuration Management Software 17

II Frame of Reference 19

3 Discrete-Event Simulation 21
3.1 Simulation and Models 21
3.2 Simulation Usage 23
3.3 When is Simulation Appropriate 24
3.4 Problems and Opportunities 25
3.5 The Simulation Study Realization Process 27
3.6 Simulation Development Time 27
3.7 Simulation Input Data 30
3.8 Discrete-Event Simulation Tools 32
3.9 Discussion and Conclusions 32

4 Discrete-Event Simulation and Integration 33
4.1 Integration Types 34
4.2 Information Integration 36
4.2.1 STEP 37
4.2.2 Using STEP for Manufacturing System Data 37
4.2.3 Exchanging and Sharing Discrete-Event Simulation Models 38
4.2.4 Tool Kit Technology 39
4.2.5 Information Integration without Standards 40
4.3 Transformation Integration 40
4.4 Application Integration 43
4.4.1 Simulation Integrated with Shop Floor Control Systems 44
4.5 Discussion and Conclusions 46

5 Software Configuration Management and Product Data Management 49
5.1 Similarities and Differences 49
5.1.1 System Architecture 50
5.1.2 Product Model 50
5.1.3 Evolution Model 50
5.1.4 Process Model 51
5.2 Software Configuration Management 51
5.2.1 Software Configuration Management Definition 52
5.2.2 The Repository and Sandboxes 53
5.2.3 Group Awareness 53
5.2.4 Configuration Save Mechanism 53
5.2.5 Managing the System Life-Cycle 54
5.2.6 The Configuration Control Board 54
5.3 Discussion and Conclusions 55

III Case Studies **57**

6 The Profilgruppen Case Study **59**
6.1 The Studied System 59
6.2 Results 61
6.3 Discussion and Conclusions 61

7 The ABB and Volvo Case Study **63**
7.1 Input Data Source 65
7.2 Input Data Structure 65
7.3 Stop Codes 66
7.4 Input Data Analysis 66
7.5 Managing a Family of Models 68
7.6 Results 70
7.7 The Scania Data Collection System 71
7.8 Discussion and Conclusions 71

8 The First BT Products Case Study **73**
8.1 Configuration Management 73
8.2 The Model 75
8.3 Communication Channels 75
8.4 Documentation 76
8.5 Results 76
8.6 Discussion and Conclusions 77

9 The Second BT Products Case Study **79**
9.1 System Description 79
9.2 Verification and Validation 80
9.3 Experiments 81
9.4 Results 81
9.5 Discussion and Conclusions 81

IV Contributions **83**

10 Discrete-Event Simulation Methodology **85**
10.1 Capability Maturity Model 85
10.2 Process Maturity Levels 86
10.3 Key Process Areas 88
10.3.1 Level 2 Key Process Areas 88
10.3.2 Level 3 Key Process Areas 92
10.4 Simulation Study Participant Roles 94
10.5 Simulation Model Properties 94
10.6 Simulation Study Properties 95
10.7 Simulation Methodology Properties 97
10.8 Documentation 97

10.9 Verification and Validation 98
10.10 Simulation Study Phases 98
10.10.1 The Functional Phase 99
10.10.2 The Conceptual Phase 102
10.10.3 The Design Phase 105
10.10.4 The Realization Phase 107
10.10.5 The Experimental Phase 108
10.10.6 The Implementation Phase 109
10.10.7 The Review and Learning Phase 109
10.11 Discussion and Conclusions 109

11 Documenting Discrete-Event Simulation Studies 113
11.1 Documentation Method Properties 114
11.2 The CodeDoc Tool 115
11.3 Discussion and Conclusions 118

12 Data Collection Framework 119
12.1 Data Usage 119
12.2 Dependencies 121
12.3 Losses 121
12.4 Time-Scales 122
12.4.1 Notations Used 123
12.4.2 Wall-Clock Time 125
12.4.3 Operational Time 125
12.4.4 Operating Time 125
12.4.5 Busy Time 125
12.4.6 Process Time 126
12.5 Safety Zones 126
12.6 Database 127
12.7 Input Data Generalization 127
12.8 Discussion and Conclusions 129

13 Parameterized Simulation Models 131
13.1 Components 131
13.1.1 Spreadsheet 131
13.1.2 General Logic 132
13.2 Synchronized Transport Logic 134
13.3 Modeling Method 136
13.4 Discussion and Conclusions 138

14 Integrated Factory Simulation 139
14.1 Planned Usage 139
14.2 Requirements 140
14.3 Model Building 141
14.4 Used Software 141
14.5 Developed Software 142

14.5.1 The QDBC Server 142
14.5.2 The QDBC Client Routines 142
14.5.3 The SQL Routines 142
14.6 The Database 143
14.7 The Simulation Model Objects 144
14.7.1 The Orders 144
14.7.2 The Scheduler 144
14.7.3 The Resources 145
14.7.4 The Terminating Element 145
14.8 Running the Simulation 145
14.9 Discussion and Conclusions 146

V Epilogue 149

15 Discussion 151
15.1 Comparison Between CAD, RS and DES 151
15.2 Problems 152
15.2.1 Cost and Benefits 152
15.2.2 A Defense of Discrete-Event Simulation 154
15.3 Contributions 156
15.4 Construct Validity 157
15.5 Internal Validity 157
15.6 External Validity 158
15.7 Reliability 159
15.8 Applicability 159

16 Conclusions 161

17 Future Research 163

Bibliography 180

A Proposed Discrete-Event Simulation Process 181

B Availability Related Definitions 211
B.1 Basics 211
B.2 Availability 212
B.2.1 Availability Based on the Time Interval 212
B.2.2 Availability Based on Downtime Type 213

C An Example of the Incremental Development of Documents 215

List of Figures 219

List of Tables 221

Acronyms 223

Glossary 229

Part I

Prologue

Chapter 1

Context

Dynamics and flexibility are of increasing importance. One of the most crucial factors in achieving dynamics and flexibility is time management. A rapid response to market demands is the key to high competitiveness. The primary driving force has been an increasing globalization of markets. Speed in product development and development of the corresponding manufacturing system are key factors to adopt to the new dynamic market requirements (Järneteg 1995, Stalk & Hout 1990).

The lead-time for the integrated development of products and manufacturing systems is a vital issue. The life cycle of a product has become shorter and so has the life cycle of the corresponding manufacturing systems which makes short development time and commissioning important.

1.1 Computer-Aided Manufacturing System Engineering

In the past, manufacturing companies have used many types of unrelated simulation systems. Mechanical engineers use mechanical models such as FEM (Finite Element Analysis) while industrial engineers use DES (Discrete-Event Simulation) to e.g. simulate throughput and detect bottlenecks or RS (Robot Simulation) to verify robot cells. Furthermore, the software tools used for manufacturing system development has not been far as developed as their counterparts in the production operations and product design areas, shown in Figure 1.1 (Andersson 1997, Bolmsjö, Lorentzon & Randell 1999, Freedman 1999, Klingstam & Gullander 1997, McLean 1993).

The simulation industry has now matured to the point that almost all manufacturing processes and flow of materials through the enterprise can be modeled and simulated. The difficulty is now collecting and managing the product, process and resource knowledge base across the enterprise and to maintain configuration management of this common, shared data repository (Freedman 1999).

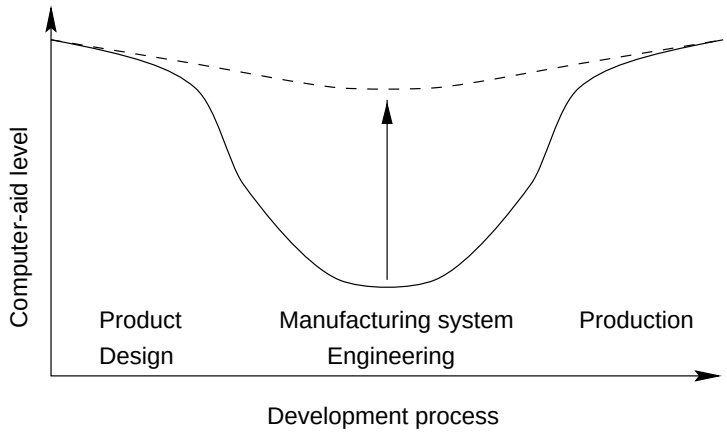


Figure 1.1: *Reformation of the product realization process. Adopted from Andersson (1997).*

McLean (1993) states that the process used to engineer, or re-engineer, manufacturing systems is often ad hoc. Computerized tools especially are used on a limited basis. McLean defines CAMSE (Computer-Aided Manufacturing System Engineering) as:

> the use of computerized tools in the application of scientific and engineering methods to the problem of the design and implementation of manufacturing systems.

The tools needed for CAMSE are complex since they should make available information used in a number of domains.

1.2 Integrated Concurrent Engineering

Time requirements for the manufacturing system development process can basically be reduced in three ways:

Integration of tools Using new and more efficient tools can reduce time and lead-time requirements. However, to gain significant time reductions the tools have to be integrated. Currently most computer systems and tools live lives on their own. The same information or logic is placed in different systems, often in different formats or with different interfaces (McLean 1993, Parks, Koonce, Rabelo, Judd & Sauter 1994).

Integration of processes Integration of the product development and the manufacturing system development processes reduce time spent on rework and quality issues, i.e. reducing the amount of redundant work (Starbek, Kušar & Jenko 1999).

Parallel development To shorten time to market it is necessary to cut sequential activities to a minimum and introduce methods, models and tools which supports concurrent engineering. Allowing more people work in parallel will reduce the lead-time but not necessarily the total time consumption. Parallel work can be performed at the document level, i.e. the same document can be edited by several persons at the same time, or at the activity level, i.e. concurrent engineering.

In the work presented, the focus has been on the two latter strategies.

Evaluation of a manufacturing system at an early stage in the product realization process is important for three reasons. Firstly, the planned manufacturing system can be quantitatively evaluated in terms of throughput, cost etc. Secondly, different manufacturing system designs can be evaluated and compared. Thirdly, different product design solutions can be evaluated with regard to the manufacturing system and the possible production processes.

In a survey performed by Institute for Defense Analyses, CE (Concurrent Engineering) was found to have tangible benefits. Where applied properly, CE resulted in higher utility to the user, higher quality and lower product cost with less development time. For example (Winner et al. 1988):

- Development and production lead-times:
  - Product development time shortened as much as 60%
  - Production time shortened 10%
  - Total process time shortened as much as 46%
- Measurable quality improvements:
  - Manufacturing defects reduced as much as 87%
  - Yield improvements as much as 400%
  - Field failure rates lowered as much as 85%
- Engineering process improvements:
  - Engineering changes/drawings reduces as much as 93%
  - Early production engineering changes reduced 50%
  - Inventory items stocked reduced as much as 60%
  - Engineering prototype builds reduced as much as 66%
  - Scrap and rework reduced as much as 87%

CE focuses on three major concepts (Hoffman 1998):

1. The integrated product-development process must be both understood and modeled well enough to be repeatable, ensuring systematic success;
2. All relevant perspectives, from customer requirements through internal constraints, must be considered in defining and designing the product;

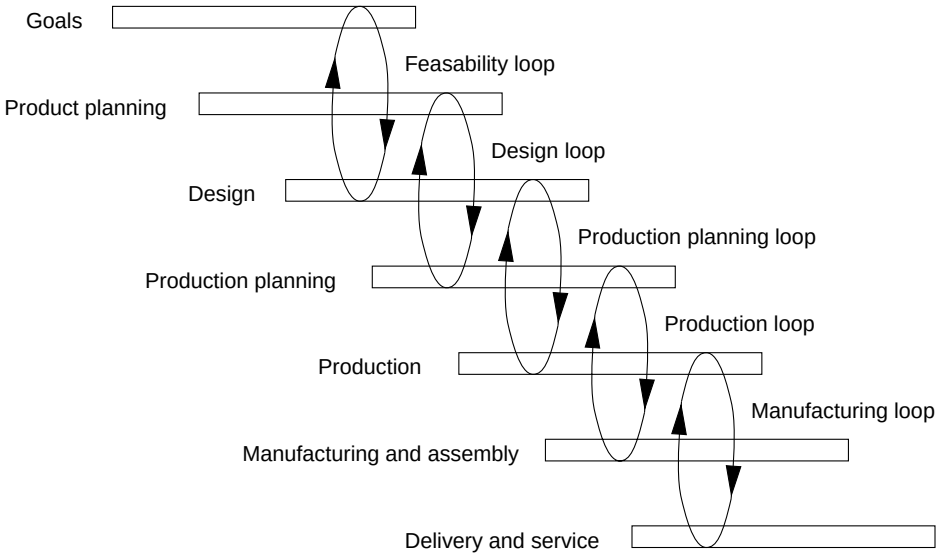


Figure 1.2: *Implementation of concurrent engineering with 3-T loops (Prasad 1996).*

3. All perspectives must be integrated to yield a global optimum, i.e., a cost-effective, robust design that is tolerant of manufacturing and use variations.

Integration of the business processes by using advanced simulation software tools is expected to speed up the total development process and at the same time produce information of higher quality due to the fact that it enables more people to take part earlier in the development work. As much as 60–80 percent of the total manufacturing cost might be affected by the decisions made in the early phases of product design (Kusiak & Lee 1996).

Prasad (1996) views the product realization process as seven groups of activities, shown in Figure 1.2. Starbek et al. (1999) defines the level of concurrency in a product realization process and it is suggested that the so called 3-T loop be selected which means that there are interactions among three activities as seen in Figure 1.2. Bolmsjö & Gustafsson (1998) and Gustafsson (1998) have presented a similar view, shown in Figure 1.3.

The integration presented so far is with the information sources needed to design the product and the corresponding processes and manufacturing system. A DES model also requires system logic and how the system interacts with surrounding systems as well. System logic and interactions with surrounding systems includes planning and scheduling systems, SFCS (Shop Floor Control System)s, human interactions with the system etc., creating an even more complex picture of the information required. In Chapter 4 the concept of integration will be presented in more detail.

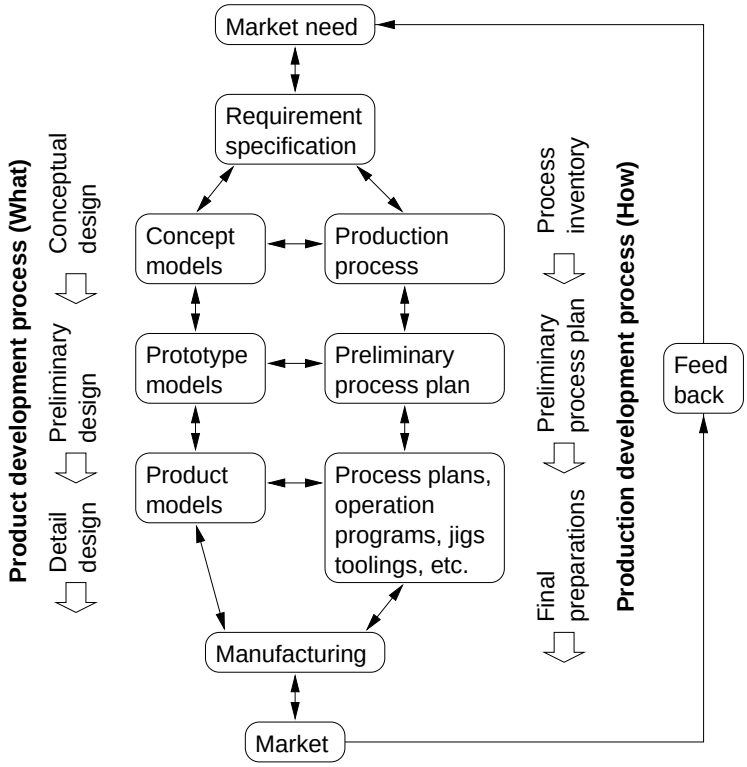


Figure 1.3: *Parallel development process in the context of product and manufacturing system development (Bolmsjö and Gustafsson 1998, Gustafsson 1998).*

1.3 Research Problem

It is believed that if the lead-time of the simulation study can be reduced it will facilitate the integration of the DES process in the product realization process. Reducing the lead-time makes it possible to actively use DES as a design tool. The reduction of lead-time also moves the detailed manufacturing system analysis forward in time, thus gives more time for pre-analyzes and reduces the amount of rework otherwise needed.

Research problem 1 *Time consumption and lead-time reduction*
How can the time consumption and lead-time of a DES study be reduced in the context of the manufacturing system (re)design process and what would such a DES study process look like?

1.4 Research Objective

The presented objectives below have the overall objective to reduce the total development lead-time with a DES study life cycle view. Another overall objective with the presented research objectives is to facilitate the integration of simulation activities in the product realization process. However, I do not intend to make that a research objective of it's own since that is quite another issue.

In Chapters 10 and 11 a methodology is presented that has the following objectives.

Research objective 1 *Discrete-event simulation process*
Develop a DES process that reduces the time consumption and lead-time of the simulation study.

In Chapter 12 a framework for collecting input data for several usages is presented.

Research objective 2 *Input data quality*
Develop a framework for collecting, analyzing, and using input data with a life cycle approach. Input data should be usable for DES, online monitoring of manufacturing system performance, retrospective manufacturing system performance analysis, and continuous improvements of the manufacturing system.

During the case studies performed at VCBC (Volvo Car Body Components) it became apparent that the development time of the simulation models needed to be very short. Therefore an added effort was made to decrease the development time even more for the type of manufacturing systems used at VCBC. The method is presented in Chapter 13.

Research objective 3 *Decreased simulation model development time for subassembly lines*

Develop a method for fast DES model development without sacrificing simulation model accuracy. The scope is subassembly lines where synchronization of manufacturing cell activities is performed.

Randell, Holst & Bolmsjö (1999) presents an approach for building large simulation models. The approach has limits and in Chapter 14 a method for integrated factory simulations is presented.

Research objective 4 *Integrated factory simulation*

Develop a method based on the usage of existing databases to build and execute DES models.

Chapter 2

Materials and Methods

This chapter is devoted to the scientific method chosen and the tools used.

It was decided early that the research would involve case studies to be able to observe the inner details of the methods used when performing DES studies. How is then research performed when performing case studies? The classical scientific methods for natural sciences do not seem to apply with their focus on theory and experiments. To what extent can a case study be considered an experiment? Not at all I would say, since in an experiment the parameters known to affect the outcome of the experiment is kept under control. This is not possible when performing case studies. The researcher is very much dependent of the environment, and thus the parameters, in which the case study is performed. That is, only to a very limited extent can the variables that might affect the outcome be kept under control.

Natural sciences to a large extent try to explain phenomena with theories and use experiments to verify those theories or discover phenomena during experiments and tries to develop theories for those phenomena. I will not dwell on the philosophical debate whether science is theory driven or driven by experiments, which is much better covered by e.g. Chalmers (1999) and Hacking (1983). Instead the focus is on the fact that experiments in the sense physicians or chemists use can not be performed for the kind of research presented here for the reasons presented above. Furthermore, such experiments focus on what can be referred to as *natural laws*. I doubt that there is such a thing as a natural law for performing DES studies. There are so many parameters affecting a study that it is impossible to control or even identify all of them. However, as will be presented in Chapter 10, it is possible reduce the variance of the DES study result by defining and enforcing polices, standards, and organizational structures.

2.1 Scientific Method

In general the methods of scientific work will help to produce a more reliable knowledge than the knowledge gained by *just living and learning*. These methods should help to access control of how experiences and values influence the knowledge gained, correct information and enough information about the phenomena under study, high quality information, and a structure in the information. Furthermore, the methods should help put together and express the information in an understandable pattern (Patel & Tebelius 1987).

Research can go along two alternative roads; *deduction* or *induction* (Patel & Tebelius 1987). Inductive arguments proceed from a finite number of specific facts to a general conclusion (Chalmers 1999). The research performed has been explorative based on individual cases and can thus be classified as inductive. When inductive research is performed, the method is not that much at focus, instead the scientific quality is determined by the patterns found in the information collected (Patel & Tebelius 1987).

According to Chalmers there is a set of conditions that has to be satisfied if inductive inference from observable facts should be transformed into laws. Firstly, the number of observations forming the basis of a generalization must be large. Secondly, the observation must be repeated under a wide variety of conditions. Thirdly, no accepted observation statement should conflict with the derived law. According to Yin (1994) these conditions can be relaxed for case study research.

2.2 Research Type

Patel & Tebelius defines applied research as a search for new knowledge with a specific application in mind, which is a suitable description of the research performed. The specific application in this case then being development of DES models.

The way the research problem is described decides if the research should be quantitative or qualitative. In this case qualitative research have been performed in that there has been a search for knowledge that should investigate, interpret and understand phenomena.

In qualitative research the problem should be studied historically and how different explanations affect the way the problem is to be handled. The problem must be understood in a wider perspective since the researcher's comprehension is the main tool to gather information (Patel & Tebelius 1987).

It would have been desirable to quantify some of the results from the case studies to be able to make comparisons between different methods. However, such comparisons are not possible to make due to the intrinsic repeatability problem of the research performed. Comparison of the contributions with traditional methodologies therefore had to be estimated by the researcher, sometimes together with others, which bring us to the question of objectivity

and subjectivity.

2.2.1 Objectivity and Subjectivity

How the objectivity and subjectivity issue is dealt with is related to the scientific viewpoint as well as the research problem.

The qualitative research is based on the postulation that we can gain access to each other's inner worlds through the language. This means that the within perspective is a presumption for the research process (Patel & Tebelius 1987). This is a difficult part and great care must be taken when interpreting other peoples' words. The quality of the research is therefore very much the responsibility of the researcher.

2.2.2 Participant-Observation

Participant-observation is a special mode of observation where the researcher is not merely a passive observer. Instead, the researcher may assume a variety of roles within a case study situation and may actually participate in the events being studied (Yin 1994).

To be able to see all the inner details of the problems this work has been performed with participation. The main advantage with participant-observation is the possibility to study events in their context at the same time as they happen. When participating, observations and conclusions can continuously be fed back and minor events can be manipulated. The mere observer just observe and conclude later (Patel & Tebelius 1987, Yin 1994).

The most distinctive opportunity is related to the ability to gain access to events or groups that are otherwise inaccessible to scientific investigation (Yin 1994). It would be difficult to gain access to the case studies without participation, i.e. the input of work and knowledge by the researchers. The tasks performed in the case studies were not common knowledge and were therefore often partly conducted by the team of researchers in the projects.

Another distinctive opportunity of participant research is the ability to perceive reality from the viewpoint of someone inside the case study rather than external to it.

The major problem related to participant observation is the potential biases produced (Becker 1958). Another disadvantage is that observations, and especially participant observation, require too much attention and is expensive (Patel & Tebelius 1987, Yin 1994).

2.2.3 Interviews

Interviews are an intrinsic part of a simulation study, thus questions were posed both for the purpose of completing the simulation study as well as for the research.

The interviews performed during the case studies were *unstructured* and without *standardization*. Since different people in the projects had different knowledge and background, it was not possible to standardize the questions. To maximize the amount of information collected the interviews were unstructured.

2.3 Case Studies

Case studies are the preferred strategy when *how* or *why* questions are being posed, when the investigator has little control over events, and when the focus is on a contemporary phenomenon within some real-life context (Patel & Tebelius 1987, Yin 1994).

Case studies are generalizable to theoretical propositions and not to populations or universes. The case study does not represent a sample and the goal is to expand and generalize theories, i.e. *analytic generalization*, and not to enumerate frequencies, i.e. *statistical generalization* (Yin 1994).

A strength of the case study is its *ecological validity*, i.e. the possibility of generalizing from one context to another. A weakness is of course the *population validity*, i.e. the possibility to generalize to a larger group (Patel & Tebelius 1987).

Case studies have a holistic perspective and tries to cover all thinkable aspects. That means that case studies cover more variables on a smaller population than survey studies do (Patel & Tebelius 1987).

Case studies can deal with a full variety of evidence: documents, artifacts, interviews, and observations. Moreover, in some situations, such as participant-observation, informal manipulation can occur (Yin 1994).

2.3.1 Single or Multiple Case Study Design

Yin (1994) gives a number of reasons for using single case studies. The most appropriate in this context is in which the case represents an extreme or unique case. However, there were common topics in the performed cases and the cases performed can therefore be said to be both single and multiple depending on topic. The choice between single and multiple case designs remains within the same methodological framework and no broad distinction is made between.

2.3.2 Group to Research on

It has been natural to participate in, or observe real projects in the manufacturing industry and in particular in the industries working actively with questions related to the research performed.

Close cooperation has been established with industries, which has secured an industrial relevance throughout the research work. The case studies have

been performed to develop and apply methods in the industry. The case studies have thus enabled dissemination of results to the industry, fed back experiences of industrial relevance, and provided information that enabled refinements of methods for practical use.

Several case studies are presented in this thesis. Two case studies were performed at BT Products in Mjölby. ABB Body-in-White and Volvo Car Body Components in Olofström were the partners when performing a case studies at Volvo Car Body Components. Another case has been performed at Profilgruppen in Åseda. Bachelor thesis cases performed at Volvo Articulated Haulers in Braås (Thuresson & Husberg 1997, Kubiak & Tsambasopoulos 1998) and at Urshults Werkstads AB in Urshult (Abrahamsson & Törnblad 1998) have been supervised. A case performed by Prosolvia has been observed at Volvo Articulated Haulers and lately, yet another project has been performed at Volvo Articulated Haulers together with Tobias Johansson and Martina Gustavsson at Teknikcentrum Kronoberg. Two Master's thesis cases have been supervised. The first performed at Enercon Windtower Production in Malmö (Solding 2001). The second at Volvo Car Body Components in Olofström (Pålsson & Quist 2002) was supervised together with Lars Holst at Division of Robotics, Department of Mechanical Engineering, Lund University.

Quite a few hours on my account was spent in the above mentioned projects. The cases at BT Products consumed at least 600 hours. The Profilgruppen case consumed some 600 hours. The ABB Body-in-White and Volvo Car Body Components case consumed 700 hours in the first stage. The second stage involved the development of new modeling methods and consumed some 100 hours. The supervision of cases has consumed 400 hours. Developing an enterprise simulator (Randell & Bolmsjö 2001) took about 200 hours. The last project at Volvo Articulated Haulers consumed another 100 hours. In total, approximately 2700 hours have been spent performing real world simulation projects. Although not all the cases are mentioned explicitly, experiences from all the cases has been an important foundation for the research.

2.4 Criteria for Judging the Quality of Research Designs

Concepts that have been offered for tests include trustworthiness, credibility, confirmability, and data dependability (Yin 1994, *Case study evaluations* 1990). The four tests are:

Construct validity , i.e. establishing correct operational measures for the concepts being studied;

Internal validity , i.e. for explanatory or casual studies establishing a causal relationship, whereby certain conditions are shown to lead to other conditions, as distinguished from spurious relationships;

External validity , i.e. establishing the domain to which a study's findings can be generalized

Reliability , i.e. demonstrating that the operations of a study, such as the data collection procedures, can be repeated with the same results.

2.4.1 Construct Validity

The construct validity test is especially problematic in case study research. Case studies are often criticized for the fact that a case study investigator fails to develop a sufficiently operational set of measures and that subjective judgments are used to collect the data. To meet the test of construct validity, an investigator must be sure to select the specific types of changes that are to be studied in relation to the original objectives of he study, and demonstrate that the selected measures of these changes do indeed reflect the specific types of change that have been selected.

Three tactics are available to increase construct validity. The first is the use of multiple sources of evidence. A second tactic is to establish a chain of evidence. The third tactic is to have the draft case study report reviewed by key informants (Yin 1994).

2.4.2 Internal Validity

Although internal validity is a concern only for causal or explanatory case studies, in which an investigator is trying to determine whether event x led to event y, I would say the problem exists in this case as well. If the investigator incorrectly concludes that there is a causal relationship between x and y without knowing that some third factor, z, may actually have caused y, the research design has failed to deal with some threat to internal validity. As pointed out above, this is a problem in the research performed.

2.4.3 External Validity

The third test deals with the problem of knowing whether a study's findings are generalizable beyond the immediate case study. Critics typically state that single cases offer a poor basis for generalizing. These critics tend to see cases with statistics in mind, i.e. analytic generalization. However generalization is not automatic. A theory must be tested through replications of the findings in a second or even a third case. Once such replication has been made, the results might be accepted for a much larger number (Yin 1994).

2.4.4 Reliability

The objective of the reliability test is to be sure that, if a later investigator followed exactly the same procedures as described by an earlier investigator and

conducted the same case study all over again, the later investigator should arrive at the same findings and conclusions. The goal of reliability is to minimize the errors and biases in a study. Note that it is the same case study that should be performed again, not another case study replicating the first (Yin 1994).

2.5 Discrete-Event Simulation Software

In the performed case studies QUEST (Queuing Event Simulation Tool) has been used. QUEST is a DES system from DELMIA (Digital Enterprise Lean Manufacturing Interactive Applications) with integrated 3D capabilities. Custom logic was built using SCL (Simulation Control Language), the simulation programming language in QUEST. The BCL (Batch Control Language) in QUEST was used to run experiments, dynamically alter models and build models.

2.6 Configuration Management Software

CVS (Concurrent Versions System) has been used for CM (Configuration Management) and then not only for the simulation studies, but for almost all documents generated to perform the presented research. CVS is a SCM (Software Configuration Management) system allowing concurrent development on the document level in heterogeneous and distributed environments. To utilize repository access via Internet a server was setup on an IRIX workstation at Lund University. SCM and PDM (Product Data Management) are described in more detail in Chapter 5.

Part II

Frame of Reference

Chapter 3

Discrete-Event Simulation

In this chapter simulation, and especially DES, will be presented. The presentation has been kept as short as possible, since the domain is well covered in the literature.

3.1 Simulation and Models

The Oxford English Dictionary describes *simulation* as:

> The technique of imitating the behavior of some situation or system (economic, mechanical, etc.) by means of an analogous model, situation, or apparatus, either to gain information more conveniently or to train personnel.

Or put in another way, simulation is the technique of building a model of a real or proposed system so that the behavior of the system under specific conditions may be studied (Ball 1996).

Simulation is a quite general term and there is plenty of room for misconceptions. Spreadsheet packages of today gives the possibility to generate *what-if* scenarios quite easily (Carrie 1988). Ordinary programming languages and tools like Matlab also makes simulation possible for almost anyone to perform. Then there are all the dedicated simulation tools for e.g. FEM, RS, ergonomics, etc.

The definition of a *model* given in The Oxford English Dictionary is:

> A simplified or idealized description of a system, situation, or process, often in mathematical terms, devised to facilitate calculations and predictions.

Mathematical models are the vast majority of models. Mathematical models represent a system in terms of logical and quantitative relationships that

are then manipulated and changed to see how the model reacts, and thus how the system would react provided the model is valid (Law & Kelton 1991).

Analytical solutions are exact solutions to the mathematical model. However, the majority of real world applications prove to be too complex requiring vast computing resources. In those cases, the model must be studied by means of a simulation, i.e. numerically exercising the model for the inputs in question to see how they affect the output measures.

Models can be classified along three dimensions as (Banks, Carson & Nelson 1996, Law & Kelton 1991):

- static or dynamic
- deterministic or stochastic
- discrete or continuous

A *static* simulation model is a representation of a system at a particular point of time. A *dynamic* simulation model represents a system over time, i.e. the system state, entity attributes and the number of active entities, the contents of sets, and the activities and delays currently in progress are all functions of time and are constantly changing over time.

If a model contain no probabilistic components it is *deterministic*, i.e. the result is always the same given the same input. In a *stochastic* simulation model the behavior is determined by stochastic variables. In the real world most things are in fact stochastic. Typical stochastic variables are cycle times, time between failure and repair times.

A *discrete system* is one in which the state variables change at a discrete set of times. A *continuous system* is one in which the state variables change continuously over time.

The scope of this text will be dynamic, stochastic, and discrete simulation models. When not otherwise stated the term *simulation* will designate discrete-event system simulation of dynamic and stochastic systems.

Banks et al. (1996) has the following definition of a *system*.

> A *system* is defined as a group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose.

A system is often affected by the *system environment*. It is therefore necessary to decide on the *boundary* between the system and its environment (Banks et al. 1996). The boundary is highly dependent of the purpose of the simulation study.

Decomposition of a system into its entities simplifies the analysis. Even though the studied system is large and complex, the entities are seldom complex. A logical model of a complex system may be built by incorporating a number of simple relationships. With these simple relationships it is possible to predict the behavior of the entire system. The more complex a system

becomes the more simulation is preferred over theoretical equations (Carrie 1988).

Strictly we gain information only about certain aspects when simulating. The aspects that has not been modeled can not be studied (Ball 1996, Carrie 1988). This is a fundamental point about simulators, each is designed to serve some purposes. The definition, again from The Oxford English Dictionary, of a *simulator* is:

> An apparatus for reproducing the behavior of some situation or system; esp. one that is fitted with the controls of an aircraft, motor vehicle, etc., and gives the illusion to an operator of behaving like the real thing.

3.2 Simulation Usage

Based on the number of published papers from the USA in recent years WSC (Winter Simulation Conference)s (Benjamin, Erraguntla & Mayer 1998, Farrington, Black Nembhard, Sturrock & Evans 1999, Joines & Barton 2000, Peters & Smith 2001), one can draw the conclusion that the USA still leads the field both in DES related research and industrial applications. Regarding the use of simulation in Sweden and other parts of the world a few statistics have been found (Holst, Randell & Bolmsjö 2000*b*, Holst & Bolmsjö 2001).

Umeda & Jones (1997) have studied DES usage in Japan over time. The conclusion is that the use of simulation in Japanese industry is still modest although no figures are given. The use of simulation is predicted to increase.

In a 1997 survey performed by Jackson (1998) of 64 Swedish industrial companies of various size, 54% were using simulation and 31% were using DES. The survey also showed that 70% of the users only used DES in isolated projects.

In a survey performed 1999 by Eriksson (1999) on the use of continuous simulation and DES in Swedish industry. Only 5% of the companies used simulation frequently, see Table 3.1. Furthermore, as few as 8% considered their competence regarding simulation to be adequate, which indicate that there is generally a low level of theoretical simulation knowledge.

An extensive survey on simulation usage in Germany is presented by Heitmann, Hirschberg, Rauh & Wunderlich (1997) and Hirschberg & Heitmann (1997). Simulation users in this case were mainly those using FEM, graphic 3D simulation (kinematic) and DES. Simulation usage in Germany, shown in Table 3.2, differs significantly from the usage in Sweden. However, the term simulation has in this case a wider definition, shown in Table 3.3. Still the usage of e.g. DES seems to differ significantly from the usage in Sweden.

3.3 When is Simulation Appropriate

The main motives for simulation is well put by Rooks (1997).

Table 3.1: *Results from the survey performed by Eriksson. The survey was sent to the CEO and Production Manager of 400 companies with more than 20 employees, i.e. 800 surveys of which 155 were answered (Eriksson 1999).*

In our company simulation has	*I strongly disagree*	*I somewhat disagree*	*I agree*	*I somewhat agree*	*I strongly agree*
been frequently used	60%	16%	7%	1%	4%
been seriously considered	42%	16%	17%	5%	3%
a solid knowledge base	46%	21%	10%	5%	3%
given us good experiences	51%	15%	10%	5%	2%

Table 3.2: *Simulation usage in Germany according to 24 interviews in 15 companies and a questionnaire with 395 respondents. One third of the questionnaires were from large companies and two thirds from small to medium sized companies (Hirschberg and Heitmann 1997, Heitmann et al. 1997).*

User	*Percentage*
Current user	65%
Plan to use simulation	11%
Previous user	3%
Not a user	21%

Table 3.3: *The different simulation types used by the current users (Hirschberg and Heitmann 1997, Heitmann et al. 1997).*

Type	*Percentage*
Other	20%
Structure dynamic simulation	16%
Graphic 3D simulation (kinematic)	48%
FEM simulation	50%
DES	58%

> Simulation is a powerful tool in optimizing the design of the manufacturing system, which can be analysed, optimized and verified before the purchase or installation of any capital equipment. It is a means to avoiding costly errors, speeding up commissioning and ensuring the plant operates right first time.

Modeling and simulating systems, such as manufacturing systems, can be achieved using a number of tools and techniques one of which is DES. Simple spreadsheets are useful for quick analyzes of e.g. capacity. However, such assessments do not take into account time repercussions. Another modeling approach is that of AQNM (Analytical Queuing Network Model). These models can provide quick estimates of steady state results regarding total system output and average resource utilization and require a relatively small number of data inputs. However, unlike DES that provides transient stated results, AQNM models analyze the system under steady state conditions. AQNM models also require limiting assumptions about the system characteristics like rework, reentrant flow, and non-exponential random failures, i.e. dynamic and detailed analysis of complex systems requires the use of DES (Grewal, Bruska, Wulf & Robinson 1998).

Three typical simulation study types can be identified:

- An explorative study of an existent system to find possible improvements. The simulation model is used to rapidly make a number of changes to see if the system can be improved by, e.g. changing scheduling rules, operating rules, or the system itself.
- Study an existent system with some suggested changes and compare the results to see if the changes are profitable. This is similar to the previous but here the model is used to validate proposed changes, not to find them.
- Design and validate a system to be. In this case the simulation is used in the design process to validate the performance or function of the system, and at the same time detect possible enhancements of the proposed system.

3.4 Problems and Opportunities

There are several advantages with DES, but the technique has also a number of problems associated with it (Banks et al. 1996, Law & Kelton 1991).

Quantitative evaluation is fundamental to the assessment of a manufacturing system design. Understanding a system's behavior and the parameters that affect performance is vital in both design and operation. Many of the measures used are influenced by the time dependent behavior of the manufacturing system. Performance variation at any point in the manufacturing process will inevitably influence other parts in the manufacturing system, both

upstream and downstream. Such dynamic factors can be critical to the evaluation of many design or operations decisions (Ball & Love 1994, De Smet, Gelders & Pintelon 1997).

During a DES study many of the inner mechanisms of the system are revealed. In many cases the studies performed before modeling result in a detailed understanding of the system. Visualization of the DES then enables even more understanding of system behavior. Furthermore, a DES can be shown and explained to others in the organization. Many users of DES think of this system understanding as one of the most important results from a DES (Savén 1995, Hirschberg & Heitmann 1997)

One of the key powers of DES is the ability to model the behavior of a system as time progresses (Ball 1996). At the same time that is a disadvantage in that the results can be hard to interpret. There is no built in analysis method to interpret the simulation. The 'spreadsheet syndrome' applies to simulations as well. Simulations generate a large volume of numbers and often have a realistic animation, which tends to generate too much confidence in the results. Simulation is not an optimizing technology since it only produces estimates of a model's true characteristics. With sophisticated output data analysis, optimization can be performed (Eriksson 1997).

Simulation has flexibility in that it can handle several different types of systems. The price paid for flexibility is that simulation results depend on who performs the simulation and analysis. Model building is a daunting task that requires much training and experience. The level of detail required can be hard to define. DES has been applied to manufacturing for about 40 years (Carrie 1988, Savén 1995). However, for most of that time is has been within the province of a few specialists, remote from, the manufacturing engineers. This is very much the case today as well, although the gap is getting smaller (see e.g. Johansson, Johnsson & Eriksson 2002). The difficulty to find engineers who can build models of complex systems easily can act as a barrier to the use of simulation (Ulgen & Thomasma 1990).

The lack of theoretical knowledge about DES in the industry is probably the one of the causes of the low usage. According to Hirschberg & Heitmann (1997) the most important barriers are the high investments in software, hardware and training. Another problem, according to Hirschberg & Heitmann, is that simulation studies require more time and effort and that simulation is considered being too complex. The complexity problem is related to the competent and professional performance of the simulation study being one of the main success factors.

Collins & Watson (1993) and Ball (1995*a*) state that the limited usage is caused by the fact that DES is hard to learn and use. However, this is only one side of the problem. Firstly, the developer must understand the system, which for complex systems is quite a task. Secondly, the model should be designed with a number of reasonable approximations. Finally, the DES should be verified, validated and the results should be interpreted. All these tasks require great skill. Of course an appropriate tool would simplify the modeling task,

but not the other tasks. Similarly, the skills required by the manufacturing engineers are no panacea.

It should be noted that I do agree in that simulation tools should be made easier to use. Especially, some of the GUI (Graphical User Interface)s are not appropriate in that they only present a graphical view of the objects hiding the system logic and the relations in between objects from the user. That is, they are not really modeling tools, mere tools for building models, and there is a significant difference.

According to Ball & Love (1994), splitting the roles of model builder and model user presents a number of potential drawbacks. Firstly, incorrect assumptions could be made by the builder. Secondly, the lead-time between a change request from the user and implementation of the change might be long. Finally, the opportunity for the user to experiment with alternatives is limited.

There are integrated solutions of different modeling approaches and DES packages (see e.g. Bartolotta, McLean, Lee & Jones 1998, Bernard 2000, Bobeanu & Filip 1995, Delen, Benjamin & Erraguntla 1998, Delen, Benjamin & Erraguntla 1999, Gmilkowsky, Eckardt & Palleduhn 1997, Gmilkowsky, Eckardt & Palleduhn 1998, Lee 1999*a*, Lee 1999*b*, Moorthy 1999, Pritsker & O'Reilly 1998, Srinivasan & Jayaraman 1997, Whitman, Huff & Presley 1997). However, none of the presented modeling methods are general and exchangeable in between different simulation tools although that is the intention of the work performed by e.g. Bartolotta et al. It would be desirable with modeling languages similar to EXPRESS in the STEP (Standard for the Exchange of Product Model Data) suite.

Simulation promise to reduce development time of a new system or the time to study modifications of an existing system. However, a simulation project can consume considerable amounts of resources.

There are also a number of advantages that has no disadvantages associated. Experimenting with the system itself is often too expensive, lengthy or impossible. In a simulation it is possible to maintain better control over experimental conditions than is possible when experimenting with the system itself. Most complex systems with stochastic elements can not be described by mathematical models nor can they be evaluated analytically. DES is then the only feasible way of analyzing the system at some level of detail. Furthermore, simulation allows studies of a system over a long time period since time is compressed.

3.5 The Simulation Study Realization Process

Typical phases in a simulation study are shown in Figure 3.1.

One of the most important and difficult tasks facing a model developer is the verification and validation of the simulation model. *Verification* is concerned with building the *model right*. It is utilized in the comparison of the

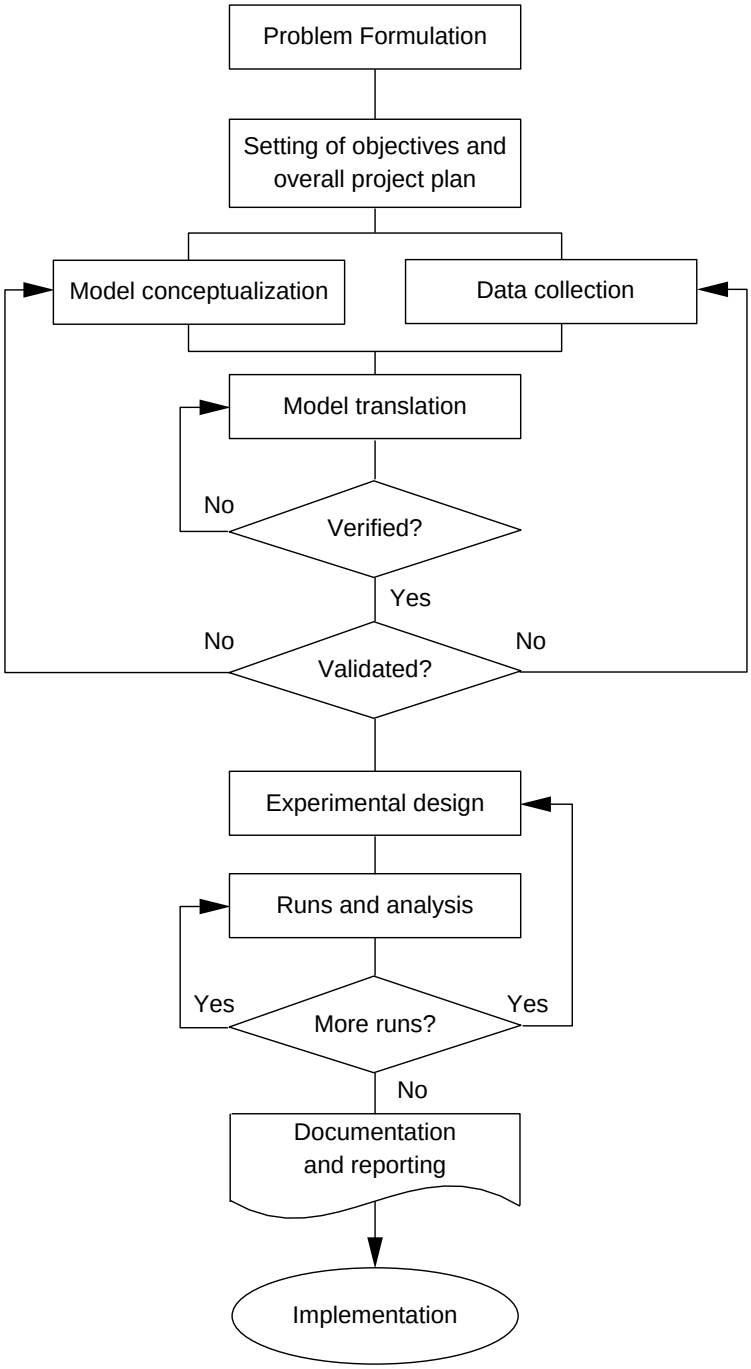


Figure 3.1: *Typical steps in a simulation study (Banks et al. 1996).*

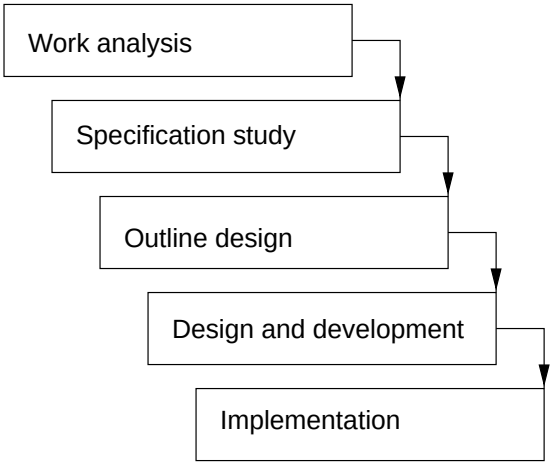


Figure 3.2: *Concepts of the waterfall methodology (Oura 2000).*

conceptual model to the computer representation that implements that conception. *Validation* is concerned with building the *right model*. It is utilized to determine that a model is an accurate representation of the real system or system to be (Balci 1998, Banks et al. 1996, Chew & Sullivan 2000, Conwell, Enright & Stutzman 2000, Sargent 2000, Sargent 2001).

The steps presented above can be compared to those of the so-called *waterfall methodology* that is the conventional approach when developing software (Oura 2000), see Figure 3.2. The steps in the waterfall methodology are similar although some steps are added for a simulation study.

The waterfall methodology works well in the case of top-down planning. This is also one of the main disadvantages of the methodology since most projects today are of concurrent nature requiring input from many people over time. The continuous changes of the specifications requires a methodology more suited for bottom-up planning (Oura 2000).

3.6 Simulation Development Time

Trybula (1994) gives a summary of the time used in the different phases of a DES study, shown in Table 3.4. Similar results are presented by Umeda & Jones (1997), shown in Table 3.5

Trybula (1994) states that the time to create a simulation model has remained constant inspite of the improvement in tools. As the tools have improved, the models have been more complex. The model development time has remained approximately constant due to increased data collection, more complexity and other modeling issues.

Table 3.4: *Time consumption in a simulation project (Trybula 1994).*

Phase	*Percentage*
Problem definition	≈ 10%
Problem analysis	≈ 10%
Data gathering and validation	10-40%
Model development	10-40%
Model verification and validation	≈ 10%
Model experiments	10-20%
Analysis of results	≈ 10%
Conclusions and recommendations	≈ 5%

Table 3.5: *Time consumption in a simulation project (Umeda and Jones 1997).*

Activity	*Percentage*	*Range*
Data collection	20	15-25
Model design	40	30-60
Animation	10	5-15
Model modification	10	5-15
Simulation experiments	10	5-20
Summary of results	10	5-10

3.7 Simulation Input Data

Simulation input data is a major problem which usually takes considerable time to collect. The simulation input data problem actually consists of a set of problems:

- availability,
- syntax and semantics,
- information model,
- dependencies, autocorrelation, and inhomogeneities,
- information content, and
- input data analysis.

Availability of input data is by several authors stated as the main problem. Many companies seem to plan and control production with simple rules of thumb, which makes correct data dispensable. Others have the data, but well hidden in their information systems. The same data can also be in several information systems, but with inconsistent values. Yet another problem is the dependencies hidden in the data.

Two problems occur as information is transferred between applications. The first problem is that of *syntax*. The second problem is that of *semantics*;

for example, 'socket' has different meanings in mechanical, electrical, and software engineering domain models (Parks et al. 1994).

It is not how input data is formatted in tables, numeric formats etc. that is the major problem. Such reformatting of data is usually quite simple to perform once the syntax problem is solved although some effort is required in automating the conversions. Instead, the major problem in the information content that tends to be focused on static means instead of the distributions desired for DES. Furthermore, dynamic losses, i.e. time in the blocked, starved, and repair states, are usually aggregated in e.g. cycle times. In conclusion, the simulation modeler has to take great care when using available input data (for a more detailed discussion on the topic see e.g. Banks 1998, Banks, Carson, Nelson & Nicol 2000, Law & Kelton 1991).

Input modeling modules can be used to display, summarize and analyze raw data and rank fitted distributions based upon the data. Not all simulation tools include input data analysis modules. In those cases the simulation engineer have to rely on other commercial products, such as ExpertFit (ExpertFit www) or Stat::Fit (Benneyan 1998, Stat::Fit www). Each can be used to fit observed data to a wide variety of distributions and export the distributions with the correct syntax to the simulation tool of choice (Swain 1999).

Trybula (1994) means, based on experience, that in reality the phases in a simulation study are not discrete and separated in time. Instead they are overlapping. One of the major drawbacks is the initial activities. The problem definition, problem analysis and data gathering become mixed together. After a short time, the pressure of delivering results or showing that something is being done cause the modeler to start gathering data. Since the modeling and data gathering phase overlap the scope of the effort constantly changes. This causes the model building to change direction and extend the time to complete. The result of this is that there is little time for verification, validation, experimentation and analysis, which result in a project that is late, incomplete and over budget.

To avoid the above mentioned problem Trybula propose a philosophy that is based on developing reasonable estimates of the possible data values and to continue on with the normal model development adding real data as it becomes available. If data is not available, the analysis can aid in analyzing possible output characteristics based on ranges of control variables. The structured approach provides a basis of minimizing delays in the model build process due to lack of data.

Simulation tools of today relies on embedded models, i.e. data and logic are embedded in the model. Drake & Smith (1996) and Peters, Smith, Curry, LaJimodiere & Drake (1996) concludes that simulations are in general generated off-line with limited direct connections to the actual data. Instead input data is gathered and analyzed outside the simulation environment. This is stated to be the primary reason for models not being reused after the initial design usage of models.

Liyanage & Perera (1998) presents seven factors that can lead to longer data collection time. The presented factors are based on a literature review and a questionnaire survey conducted at the 1997 WSC. The list below highlights many of the problems mentioned above. The seven pitfalls were ranked in the survey and they are listed in the order they were considered to be influential (1 — most influential, 7 — least influential).

1. Poor data availability.
2. High level of model details. It is stated that higher level of detail does not necessarily lead to a higher accuracy but lead to longer data collection time.
3. Difficulty in identifying available data sources. When data is available it is often located in different sources and in some cases the same data is in several sources. In the case where the same data is in several sources the data might differ due to poor integration.
4. Complexity of the system under investigation. When the system is too complex, the simulation modeler tends to identify and collect data in and ad hoc manner.
5. Lack of clear objectives makes it impossible to decide an appropriate level of detail of the simulation model.
6. Limited facilities in simulation software to organize and manipulate input data.
7. Wrong problem definition which cause the modeler to identify, collect and analyze invalid data.

3.8 Discrete-Event Simulation Tools

Overviews of software packages for simulation have been performed by e.g. ARGESIM (www), Klingstam & Gullander (1997), Nwoke & Nelson (1993), Kosturiak & Gregor (1995), Swain (1997), Swain (1999), and Johansson et al. (2002). There are also a few packages, mainly for research (Bolier & Eliën www, Fishwick & Cubert www, gpdes3d www, OMNeT++ www, ParaSol www). There is evidently a large number of tools to chose from and an evaluation of what tool to use is difficult. McHaney & White (1998) and Hlupic, Irani & Paul (1999) present theoretical frameworks for tool selection.

There are a number of ways of classifying simulation tools. For a more detailed discussion on this topic, please refer to (Ball & Love 1994, Ball 1996, Banks, Aviles, McLaughlin & Yuan 1991, Boughton 1995, Bridge 1990, Law & Kelton 1991, Love & Bridge 1988, Pidd 1992*a*, Pidd 1992*b*, Swain 1999).

Simulation has great power and flexibility and each year the tools become more powerful. The improved functionality also increases the complexity of

the tools and thus does not always make them easier to use (Ball 1995*b*, Shewchuk & Chang 1991, Swain 1999). Shewchuk & Chang therefore state that achieving both ease of use and a wide range of applications is difficult in practice.

3.9 Discussion and Conclusions

As pointed out, input data poses one of the greatest problems when using DES and therefore Chapters 12 and 14 are devoted to solve some of the input data problems. There is promising research focusing on integrating simulation with the needed information sources that will facilitate the simulation task in general in the future (see e.g. Bernard 2000, Centeno & Standridge 1993, Dufrene, Gharbi, Kieffer, Rebouche & Villeneuve 1994, Johansson 2001, Krause & Jansen 1999, Moorthy 1999, Xu & AbouRizk 1999).

Poor input data is a severe threat to the reliability of simulation results. However, it is not a reason for abandoning a simulation study and use other methods. The benefits of simulation are still there, although the results have to be interpreted with great care. The alternatives of simulation would still have to rely on the same poor input data and would thus not be superior to using simulation since they do not take into account the dynamic behavior in a system.

Related to this is when to use simulation. A basic rule is that when simple static calculations can not be made in a spreadsheet, simulation is probably a better solution. In reality this then applies to most systems that have a dynamic behavior, does not have a simple straight material flow, and produce more than one product, i.e. most manufacturing systems.

In summary, the technique has several desirable benefits that are well established. However, as discussed mainly in Section 3.4, there are a number of problems associated with the technique that keeps potential users off. In Part IV it will be shown how some of the problems can be relaxed.

Chapter 4

Discrete-Event Simulation and Integration

This chapter presents different definitions of integration and some of the research performed related to the work presented in this thesis. The term integration has many facets and the chapter is provided to put the contributions into a context. First the different aspects of integration are investigated and then a few basic classes of integration are defined. Each class of integration is then presented with a literature review.

Development of products and manufacturing systems are being done more often by geographically and temporarily distributed teams in heterogeneous software environments. There is a high level of outsourcing of both manufacturing and development. Designers are no longer merely exchanging geometric data, but more general knowledge (Szykman, Bochenek, Racz, Sriram & Senfaute 2000*a*, Szykman, Racz, Bochenek & Sriram 2000*b*, Szykman, Fenves, Keirouz & Shooter 2001).

Time is the ultimate cost driver and a rapid response to market demands is the key to high competitiveness (Järneteg 1995, Stalk & Hout 1990). Companies' competitiveness is today becoming more and more relying on effective communication and coordination between organizations, processes, activities, tools, and people. Integration promise to solve many of these issues.

Interoperability costs in the automotive supply chain alone has been estimated at one billion dollars. Assuming the interoperability costs are proportional, the US aerospace, shipbuilding, and construction machinery industries interoperability costs are estimated at 400 million dollars each (Brunnermeier & Martin 1999).

A study on the impact of using computerized tools for design and simulation of manufacturing systems shows savings of 50-85% in time and errors (Johansson 2001). Another study showed that at Ford the time of development is estimated to one third by the use of an integrated computer environment

(Östman 1998).

Simulation has mostly been used as a stand-alone tool with little or no interaction with other softwares (Bolmsjö et al. 1999, Klingstam 1999, Klingstam & Gullander 1997, McLean & Leong 1997). In recent years WSC papers with the word *integration* in the title have been quite common (Andradóttir, Healy, Withers & Nelson 1997, Benjamin et al. 1998, Farrington et al. 1999, Joines & Barton 2000). When browsing the papers it becomes apparent that integrated simulation has several meanings. This overview focuses on the integration of softwares for product development and the development of the corresponding manufacturing systems. Although most tools used in the development processes are discussed the focus will be on DES.

4.1 Integration Types

Vernadat (1996) defines integration as:

> Integration means putting together heterogeneous components to form a synergistic whole.

and states that

> Essential conditions for integration seem to rely on the free but controlled flow of information and knowledge, and the coordination of actions.

Vernadat lists different views on integration. Firstly, there are two levels of integration.

- *Loose integration* is when systems merely can exchange information with one another with no guarantee that they will interpret this information the same way.
- *Full integration* is if and only if (i) the specificities of any one of the systems are known only to the system itself, (ii) the systems contribute to a common task, and (iii) the systems share the same definition of each concept they exchange.

Secondly, a distinction is made between horizontal and vertical integration.

- *Horizontal integration* concerns physical and logical integration of business processes from product demand to product shipment.
- *Vertical integration* concerns integration between the various management level of the enterprise, i.e. decision-making integration.

Thirdly a distinction is made between intra-enterprise and inter-enterprise integration.

- *Intra-enterprise integration* is the integration of business processes internal to a given enterprise.
- *Inter-enterprise integration* is the integration of a given enterprise with business processes of other enterprises.

Vernadat continues and lists different complementary forms of integration that exists within an enterprise where each one builds on the previous one.

- *Physical system integration* concerns interconnection and data exchange by means of computer networks and communication protocols.
- *Application integration* concerns interoperability of applications on heterogeneous platforms as well as access to common shared data by the various applications.
- *Business integration* concerns integration at the enterprise level, i.e. business process coordination.

Eversheim, Bochtler, Grassler & Kolscheid (1997) classifies integration in three main categories.

- *Information-oriented integration* involves integration of tools for computerized support such as CAD (Computer Aided Design), CAPP (Computer Aided Process Planning), CAM (Computer Aided Manufacturing), CIM (Computer Integrated Manufacturing), etc.
- *Organizational-oriented integration* concerns implementation of team-oriented concepts such as SE (Simultaneous Engineering), CE, etc.
- *Procedure-oriented integration* is the use of methods and techniques for structuring the work in the development process such as QFD (Quality Function Deployment), DFMA (Design For Manufacturing and Assembly), AD (Axiomatic Design), etc.

Parks et al. (1994) means that system models can be categorized along three axes of integration: domain views, levels of abstraction and life cycle phases, shown in Figure 4.1. *Domain views* is the integration between different engineering discipline views of the system. *Levels of abstraction* is the integration between different levels of detail. *Life cycle phases* is the integration between phases of the design life cycle.

According to Parks et al., three types of integration exist among the models distributed.

- *Static translation* exists when similar tools need to exchange information, e.g. CAD tools that exchange their models.
- *Dynamic integration* is characterized by the ability to exchange state information between several tools each modeling some behavioral aspect of the complex system, e.g. distributed simulation (see e.g. Fujimoto

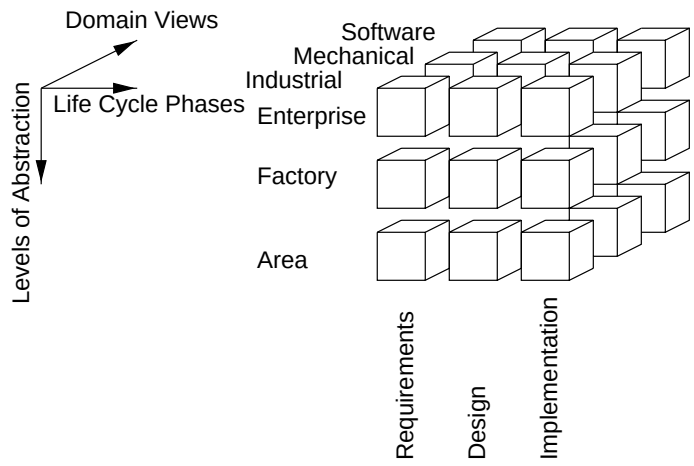


Figure 4.1: *Different modeling views of a system (Parks et al. 1994).*

1999, Norman, Tinsley, Barksdale, Wiersholm, Campbell & MacNair 1999, Prabhu & Duffie 1995, Vinoski 1997).

- *Model transformation* allows a representation of common entities in one model to be mapped or transformed into another model's representation of that same entity. Unlike static translation, these tools may be in different domains, levels of abstraction or life cycle phases, e.g. a process model is translated into a DES model (Bernard 2000).

As can be seen, there are several terms with the same or similar meanings, and in some cases they are overlapping. Here I will use the terms in the following section headings to describe different types of integration within the scope of this thesis.

4.2 Information Integration

Information integration is when information is shared/exchanged in between different applications. Information integration can be within the same domain or between several domains. The key point is that there is no information loss and that the communication thus can be bi-directional in contrast to transformation integration discussed in Section 4.3.

A common vision is that of monolithic software systems solving many of the integration problems. In this vision the product development process will be supported by a single integrated application suite. Since a monolithic system is intended to be as complete a solution as possible, interoperability is of less importance. Monolithic systems tend to be expensive and thus too expensive for small and medium sized enterprises, usually being a substantial

part of the supply chain. Furthermore, the number of applications is limited thus resulting in suboptimal solutions (Szykman et al. 2001).

To support geographically and temporarily distributed teams in heterogeneous software environments a *design repository* concept has been developed at NIST (National Institute of Standards and Technology). A design repository is an intelligent knowledge-based design modeling system used to facilitate the representation, capture, sharing, and reuse of corporate design knowledge (Szykman et al. 2000*a*, Szykman et al. 2000*b*, Szykman et al. 2001).

4.2.1 STEP

STEP was originally designed as a successor of exchange standards such as IGES (Initial Graphics Exchange Specification), SET, and VDA-FS although the scope is much wider today. Tests have shown that STEP performs better than IGES in most cases (Brunnermeier & Martin 1999, Strub 1998, PDES 1998). The STEP standard does not only cover traditional file exchange, but have a much broader scope. The three major concepts are product data exchange, product data sharing, and product data archiving (Johansson 2001, Kemmerer 1999).

When using *product data exchange*, STEP defines the form of the data that is transferred between two applications. Each application holds its own representation of the data, i.e. a redundant copy of the data is created and the information is thus represented in multiple places and thereby put a demand for revision control of the files used. Information exchange also demands a control of what is original data and who is responsible for it and the copies. The data conforming to STEP is transitory and defined only for the purpose of exchange. The transfer is initiated by the originator, i.e. there is an information push. The content is determined by a discrete event in time, i.e. the data is only a snapshot valid for at certain point in time.

In *product data sharing*, STEP defines the information interface structure when multiple applications access and operate on the same data, potentially simultaneously. The applications do not hold the data in their own preferred form. The product data of prime interest in this case is the integrated product data and not the portions that are used by the particular product data applications. The access of data, that appears to come from a single source, is done in real-time initiated by the receiver and will be accessed upon demand. Data access levels are embedded in the protocol. The share environment puts a demand for controlling the access of the data with check-in and check-out procedures.

In *product data archiving*, STEP is used to define the interface to the archive and possibly the structure of the data itself. Archiving requires that the data conforming to STEP for exchange purposes is kept for use at some other time. This subsequent use may be through either product data exchange or product data sharing.

4.2.2 Using STEP for Manufacturing System Data

Johansson (2001) addresses the use of STEP for the manufacturing system information in an integrated model with product and process information. The manufacturing system can also be seen as a product and AP214 (Application Protocol) can be used to capture relevant design data. Thus the structure of AP214 can be used to describe both the manufacturing system and product data and also the coupling between product and manufacturing system data in the form of a process. Data that is not in the scope of AP214, e.g. NC-code, can be attached through an *external reference* mechanism (Johansson & Rosén 1998, Johansson & Rosén 1999).

4.2.3 Exchanging and Sharing Discrete-Event Simulation Models

The NIST SIMA (Systems Integration of Manufacturing Applications) project focused on providing the models, integrated framework, operating environment, common databases and interface standards for a wide variety of emerging tools and techniques for designing manufacturing processes, equipment and enterprises. The primary output of the SIMA Program is a collection of specifications called IMES (Initial Manufacturing Exchange Specifications) (Bartolotta et al. 1998, McLean & Leong 1997).

Bartolotta et al. (1998) presents a document that specifies an interface specification for the exchange, archiving and sharing of DES models of manufacturing systems. An overview is shown in Figure 4.2. The objective is to provide a neutral mechanism capable of describing input data to a DES system for manufacturing systems, independent from any particular commercial simulation system. The nature of this description makes it suitable not only for neutral file exchange between dissimilar DES systems, but also as a basis for implementing and sharing databases and archiving.

4.2.4 Tool Kit Technology

A computer-aided METK (Manufacturing Engineering ToolKit) prototype has been developed at NIST as part of the CAME (Computer-Aided Manufacturing Engineering) project. The purpose of the CAME project is to provide an integrated framework, operating environment, common databases and interface standards for manufacturing engineering software applications. The toolkit consists of COTS (Commercial Off-The-Shelf) manufacturing software applications on heterogeneous platforms housed together and the METK integrates those applications to support data sharing between the applications, see Figure 4.3 (Iuliano 1995, Iuliano & Jones 1996, Iuliano 1997, Iuliano, Jones & Feng 1997). The packages implement the following functions: cell control, machine control, cell simulation, machine simulation, routing, operations planning, scheduling, NC verification and shop-floor data collection.

The METK project is centered around the concept of an engineering data package that is thought of as a package containing all the engineering data

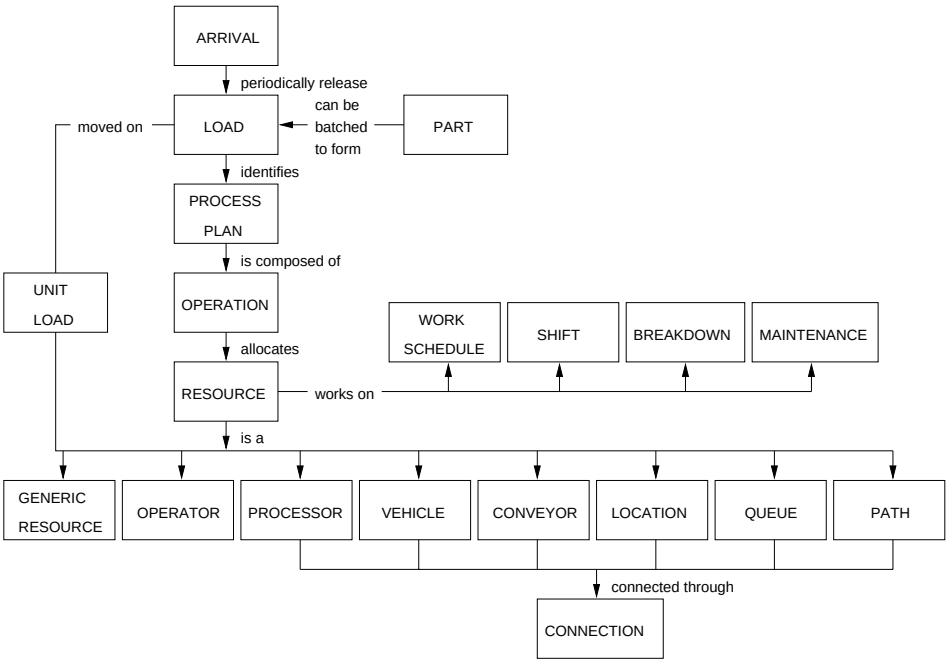


Figure 4.2: *The high level description of the requirements of the data planning model for the interface specification and the relationships between the basic data components (Bartolotta et al. 1998).*

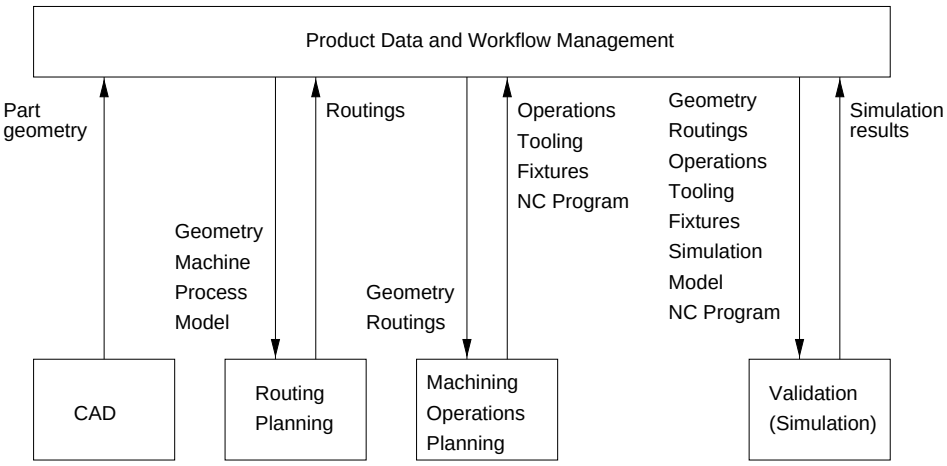


Figure 4.3: *The proposed METK system that integrates applications to support data sharing (Iuliano 1995).*

elements required for the production of a part. The engineering data elements consist of tool lists, fixture lists, NC programs, operation sheets, route sheets and geometry models.

A key point in the METK project is data validation. The integrated toolkit of software applications cross-check each other for consistency and accuracy. The QUEST simulation environment is, for instance, intended to validate the routing data in the engineering data package.

The METK was used to demonstrate that tools were commercially available to perform computer-aided manufacturing system engineering, develop a better understanding for individual engineering tools and the overall environment and identify integration standards and issues which must be addressed to implement plug-compatible environments in the future.

The goal was to have the virtual cells emulate a real factory as closely as possible. However, in the presented implementation there are some slight differences. In an integration context the most important one being that the machine controllers are embedded in the machine simulators (compare to Section 4.4.1).

4.2.5 Information Integration without Standards

Many of the previously cited authors used standards, such as STEP to facilitate integration. Parks et al. (1994) argue that in the development of a manufacturing system specialists of the different disciplines has their own methods and modeling techniques. Instead of trying to develop a single methodology or modeling approach used by every discipline, an integrated methodology, which melds the existing methods and models, can be developed. Parks et al. propose an architecture, Integrated Manufacturing Design Environment, with the characteristics:

- Coordination of design and modeling in an environment that allows designers to operate on common, familiar tools;
- Data will be stored on the local tools in the format specified by the tool vendor;
- Tools can be added and removed through an automated registration mechanism;
- Interfaces will be developed to minimize the effort that is required to attach them to existing tools;
- A message system will be incorporated to notify all concerned parties about conflicts, data updates, constraint violations etc.

With a holistic view it can be argued whether implementing and maintaining an integrated manufacturing design environment will be simpler than developing applicable standards that can be used by several enterprises and organizations.

4.3 Transformation Integration

Transformation integration is a special case of information integration where the original information is transformed to be used by another application in another domain than the original application. Parks et al. (1994) gives an example of a common problem where a simulation engineer develops a model of a manufacturing system. Many control issues, e.g. queue discipline, routings etc., are embedded in the simulation model. Other engineers need this information for their development of e.g. SFCSs. With an integrated solution the embedded information would be available and exchanged/shared among the different engineers.

McHaney (1988) identify four methods of facilitating logic transfer: *philosophic transfer*, *pseudocode transfer*, *data base transfer*, and *actual code transfer*. The methods are applied when simulation is used as a design tool. The goal of the simulation is to identify all the logic necessary to implement a controller for the system.

When philosophic transfer is applied, the simulation key ideas and assumptions are communicated textually or verbally to the system design team, which then implements this in the actual system software. One of the risks with this approach is that pieces of information is neglected, or not communicated properly resulting in a misinterpreted or omitted portion of logic which in turn will result in a discrepancy between the simulated system and the implemented real system. Philosophic transfer makes validation a more complex task. To avoid some of the problems pertaining the approach the system design team could be integrated, hence containing both simulation engineers and software engineers.

The pseudocode transfer approach is based on pseudocode generated by the simulation engineers as a method of documenting what has been implemented in the simulation model. The software engineers then analyze the pseudocode and implement it. The pseudocode transfer is more detailed thus taking into account the problem in the philosophical transfer. Validation is also facilitated. However, there is a duplication of the programming effort both to and from the pseudocode.

The database transfer approach relies on the transfer of a database developed during the simulation phase to drive the actual system. The software used has to be available for both the simulation model and the actual system controller for this approach to be applicable. The main advantage is that testing and debugging can be performed during the simulation phase and the duplicated programming effort is almost eliminated. Changes to the simulation and actual system can be done easily and consistently. Validation is facilitated as well. The disadvantage is that flexibility is lost since unusual cases and exceptions become difficult to incorporate (McHaney 1988) (compare to the database stored models described by e.g. Centeno & Standridge 1993, Weaks & Barret 1997).

In the actual code transfer approach the actual code is written in a con-

ventional language that can be called from both the actual system and the simulation model. The actual code can be implemented by either the simulation engineers or the software engineers. With this approach the duplicated effort is reduced and validation is performed for both systems simultaneously. The approach requires the simulation engineers and the software engineers to work closely. A disadvantage is that simulation engineers and software engineers has to cross the domain borders which requires knowledge in both domains. Another disadvantage is that of a more concerted effort and added debugging time initially. The actual code transfer is similar to the application integration presented in Section 4.4.1

Xu & AbouRizk (1999) presents an approach that focuses on the product, the simulated object, instead of the processes or activities. Xu & AbouRizk calls this a product-oriented simulation environment that achieves integration by adopting a product view of the constructed facility. All relevant data related to the constructed facility is stored in the *product hierarchy*. The product hierarchy represents the products' physical attributes, their relationships and the methods by which they are constructed. The *product network* is a concept used to define the construction sequence for each construction phase.

The simulation model of the project is composed by linking the product model in a given construction phase to simulation models from a library of flexible/modular models. The simulation model also represents the methods by which the project will be constructed and the process to be followed.

Moorthy (1999) have used a similar approach. FactoryCad is used within AutoCad to generate factory layouts where the objects contain simulation data stored as attributes. The basic purpose of the project was to develop a seamless automated method of generating simulation models and 3D model animations directly from CAD drawings. The simulation model data is saved in a SDX (Simulation Data Exchange) file, which is used to generate a simulation model through a translator.

The geometries are generated in FactoryCAD or other applications. Simulation relevant data is then embedded into the objects in the FactoryCAD model. Then the SDX file is generated and transformed into a simple simulation model. Additional simulation data maintained in other tools is then imported.

The point of the approach is to reduce the effort of maintaining simulation models including the geometries as the project is evolving. Continuous updates of the geometries of the simulation model are thus avoided.

Gmilkowsky et al. (1997) means that there has been a trend to integrate simulation applications in existing corporate information and communication systems. They present a simulation toolkit that is capable of generating context and granularity sensitive simulation models. The automatic generation contrasts to the component oriented systems where the modeler builds models by combining components with predefined functionality, (see e.g. Bley & Wuttke 1999).

Tatikonda & Stietz (1994) presents a methodology where the manufactur-

ing system analysis is performed in stages. In the first stage input data is collected and stored and analyzed in a spreadsheet. In the second stage a rough-cut analysis is performed using a queuing theory tool (MANUPLAN). The queuing theory simulations are used to decide on which alternative to continue to investigate in the next stage. In the next stage the queuing theory model is translated into a SIMAN model using the SimStarter program. With the SIMAN simulation model a more detailed analysis of the selected alternatives is performed. The methodology integrates a set of tools, but relies on the interoperability of that specific set of tools.

Another example of transformation integration is OLP (Off-Line Programming). Volvo's Olofström Plants estimate that from the very start, by using simulation and OLP, 20% has been gained in time of the design, manufacture, and installation of assembly equipment. 90% of the effort goes into the simulation and only 10% is needed for OLP, but 100% of the reward comes from OLP. Furthermore, the consistency of methods, syntax and improved processes, is often overlooked but can generate considerable cost savings (Rooks 1997).

It should be noted that some of the approaches presented above are not really transformation integration in that the information transformed is added to the original tool and thus redundant in that context. The original tool is thus only used to store data for the other tools.

4.4 Application Integration

Here a distinction is made between information integration and application integration. In the case of information integration the applications do not have to be aware of each other. That is, the interface between the applications is the information exchanged/shared. In the case of application integration the interface is specified for both the exchanged information and how to perform the interactions, i.e. the interaction is performed in a structured manner.

Application integration has several benefits. The logic of the integrated application does not have to be modeled twice (Ball & Love 1992, Smith, Wysk, Sturrock, Ramaswamy, Smith & Joshi 1994). The *double maintenance problem* (Babich 1986) is avoided since changes in either system will be reflected in the total system. Since it is the real application or system that is integrated with the simulation, accuracy is improved (Smith et al. 1994). Furthermore, validation of the simulation model is facilitated since parts of the modeled system is the real system (Ball, Boughton & Love 1994, Ball & Love 1992, Love & Barton 1996). The integrated application or system, e.g. a control system, can be tested prior to startup, and not during commissioning (Miles 1989). In addition, data management is simplified since the data is already present (Ball et al. 1994).

Real-time planning, scheduling, and control in conjunction with simulation is referred to as *on-line simulation* by Drake & Smith (1996). On-line simulation systems incorporate the ability to reliably predict the future behavior of the

studied system given its current status, and the ability to emulate and/or dictate the control logic of a manufacturing system.

The choice of MRP II (Manufacturing Resource Planning) policies and parameters will have an effect on the success of MRP II, and thus financial performance of the company. Love, Clarke & Gooden (1987) therefore conclude that there is a need to model the real manufacturing system and the MRP II system as a whole. Love et al. use the actual MRP II system being implemented to provide the production control system. The second element of the model consisted of a detailed simulation of the manufacturing facility.

Love & Barton (1996) present a demonstration system referred to as a WBS (Whole Business Simulator). The demonstration system includes the functions: customer demand generator, design, process planning, MRP II, accounting system, etc. The idea is that it should be possible to simulate, with a holistic view, the impact e.g. a design change will have on the overall financial performance. They give an example where a minor design change results in a different manufacturing process which, in turn, results in a better financial performance although it, at first, appears to generate higher costs.

The same holistic approach where interacting subsystems influence a systems total performance is presented by Eatock, Serrano, Giaglis & Paul (1999), Giaglis (1999), and Giaglis, Paul & O'Keefe (1999). The purpose of their approach is to propose an alternative approach to the problem of IT (Information Technology) investment evaluation.

The system under study is a BP (Business Process) with supporting IT and CN (Computer Network). The hypothesis was that there is a link between the CN and BP level with an IS (Information System) level in between. Giaglis et al. (1999) maintain that the interrelationships between BP, IT and CN are more complex than the hierarchical structure would imply. Therefore two models of the scenario were built, one that reflected the activities at the BP level and one that represented the new CN. To integrate the two models a third interface model was built, representing the IS infrastructure. Each model assessed the performance of each system. The advantage of using the three levels was that effects of changes in one system could be traced in the other systems.

Watt (1998) presents a case where several information sources and applications were integrated. Simulation was used for both off-line simulation and scheduling. Most of the information used was present in the MES (Manufacturing Execution System) and MRP (Manufacturing Resource Planning or Manufacturing Requirements Planning) systems and missing data was added.

Periodically snapshots of the fab status and static data from the MES were collected and schedules were generated by the AutoSched package. Off-line simulations were performed to test what-if scenarios and reused the same information for scheduling. New rules could be created and tested against history data. The improved rules were then applied in the scheduling system.

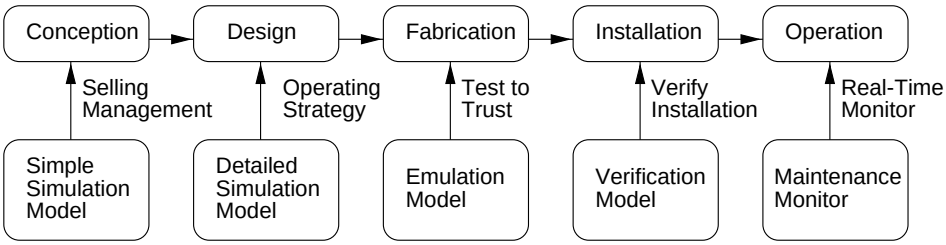


Figure 4.4: *The life cycle of an simulation automated system (Hitchens 1989).*

4.4.1 Simulation Integrated with Shop Floor Control Systems

A distinction is made between simulation and emulation. Simulation is generally applied in the early stages of a project while emulation is applied during the detailed design and implementation phases. A simulation model consists of two essential elements, the behavior of the system and the control logic. In a simulation model these two elements are usually built into the same model. In an emulation model the two elements are separated and reside on two systems. Typically the simulation model then simulates the system behavior while the control logic is run on the real control system (Hitchens 1989, Hitchens 1996).

Hitchens (1989) presents a life cycle approach to the simulation and emulation of automated systems, see Figure 4.4. The approach uses simulation in all the phases for different purposes and reuses the model from stage to stage.

In the conceptional phase the simulation is used to market a project to management.

In the design phase simulation is used to find the best solution from a set of potential designs. The focus in this phase is the overall operating strategy.

During the fabrication phase in an automation project each subsystem is built. The simulation is now connected to the real control software to test the software implementation. The controllers use the emulation model as a replacement for the physical equipment. In this way the control logic can be tested for the entire facility. Hitchens (1989) refers to this as *direct connect emulation* and states that the experience is that for every day of debug time spent using an emulator, three days in the field are saved. Direct connect emulation can reduce the cost of the software debugging task by 50-70% or more.

In the installation phase, time consumption is reduced since all vendors have tested their subsystem against a common emulation model. Hence the subsystem integration effort is reduced. The debugging effort performed in the previous phase is now increasing return on investment since time is not spent waiting for the system to be debugged.

In the operation phase the emulation is used as a diagnostic tool and runs in parallel with the operation of the physical system. If changes of the system

are required the simulation model is ready and can be used to improve the installed system or test suggested modifications before implementing changes.

Another approach to integrate simulation with SFCSs has been developed at TAMCAM (Texas A&M Computer Aided Manufacturing Laboratory) (Peters et al. 1996, Smith et al. 1994). The concepts seem the same as Hitchens at a first glance, however, there are differences.

Hitchens use the emulation model as a replacement of the physical system while the simulation is used as a MES at TAMCAM. In the TAMCAM implementation the simulation is thus an active part of the final system. In Hitchens' approach the simulation model is not an active part of the final system, but remains a passive element for monitoring.

Another difference is that in the TAMCAM approach there are direct connections to other manufacturing computer systems and databases. The simulation model, for example, is generated from the manufacturing information system (compare to e.g. Centeno & Standridge 1993, Randell & Bolmsjö 2001).

Smith et al. (1994) describe the usage for the conceptual, design and fabrication phases. They also used the simulation in look ahead mode (compare to Watt 1998).

Smith & Peters (1998) presents a paper where the focus is on the development of the simulation portion of the SFCS from a high-level system description and the use of the simulation to drive the control system and direct the activities of the FMS (Flexible Manufacturing System). The simulation level acting as a MES uses a DES for decision making. In addition to decision making, on-line simulation uses a real time link between a simulation model and the production system as a direct method of process control.

Peters et al. (1996) describes a simulation control system that is developed directly from information about the shop-floor stored in a relational database. Processing times and fallout rates are also estimated directly from external data sources.

Baudouin, Ruberti, Arékion & Kieffer (1995) presents a decision support system for a semiconductor manufacturing environment. DES fed with current fab data is used in the decision process. They have designed a LTS (Lot Tracking System) that is constructed of integrated existing computer systems and networks as well as automatic process control equipment.

To support decision making respective to scheduling or rescheduling after recovery, they used a DES interfaced to the LTS. The simulation runs allowed what-if scenarios to be tested and elaborate snapshots of the fab status. The simulation results could then be used as parameters for the scheduling system.

Judd & Abell (1996) describes an interface used to connect DES and RS packages to a PLC (Programmable Logic Controller). The interface was developed to provide an environment to develop, test and debug factory control system. From a PLC programming standpoint, there was no difference between the PLC controlling the simulation models or the actual system. As long as the simulation is written to emulate all the I/O of the actual system, the PLC

system program can be fully written, debugged and tested in the simulated environment.

4.5 Discussion and Conclusions

It has been shown that the term integration does have several meanings depending on the context. It is believed that integration in different forms is one key to reduce cost and time and thereby succeed with simulation and especially DES.

It is evident that integration is an extremely complex matter. Most authors cited here solved problems using tools within a suite of integrated tools or developed there own tools to work in a limited context.

The dynamic information, e.g. cycle time variance and failures, seems to be missing in most solutions. Another problem is the exchange of system logic. No such solution has been found. Application integration partially solves that problem, but a neutral modeling language that is capable of describing the systems logic would solve the problem of exchanging both models in between DES tools and the exchange of logic in between DES systems and e.g. SFCSs. This is related to the criticisms of the modeling tools put forward in Section 3.9. With a pessimistic view, the possibility to e.g. exchange DES models in between tools might never ever come true. The vendors do their best to protect their own interests instead of looking at the big picture and see to the customers' best interest.

In conclusion there is a large effort required to integrate the tools used in the manufacturing system development process. My view is that the integration should not be performed within monolithic suites of tools, instead standardization is required to integrate the tools. The main motive for standardization is the current interoperability cost highlighted by Brunnermeier & Martin.

Chapter 5

Software Configuration Management and Product Data Management

In the presented case studies SCM have been used for evolution control. CM is also a vital part of the methodology presented in Chapter 10. CM is well established outside the DES domain and this chapter is supplied to provide the basics.

SCM will be compared with PDM to highlight differences and similarities. The simulation tool used in the research, QUEST, can be said to lie in both domains. Information about the manufacturing process, geometries, etc. is in the PDM domain, while programming the simulation model is in the SCM domain.

In the performed research a SCM system, CVS, has been used for controlling the evolution of the simulation models and related documents. CVS was selected because it was free, easy to install, easy to learn, and available on several platforms. It also worked very well for controlling the evolution of simulation studies. PDM on the other hand seems, in general, to be complex and expensive.

5.1 Similarities and Differences

PDM is the discipline of controlling the evolution of a product design. SCM is the discipline of controlling the evolution of a software product design (Crnkovic, Persson Dahlqvist & Svensson 2001, Persson Dahlqvist, Asklund, Crnkovic, Hedin, Larsson & Ranby 2001*a*, Persson Dahlqvist, Crnkovic & Larsson 2001*b*).

The characteristics of SCM and PDM originate from the nature of the artifacts developed. In the life cycle models, PDM is focused on the hardware design phase and the production and maintenance/support phase. In the software product life cycle the development phase is usually regarded as the most

intensive part (Persson Dahlqvist et al. 2001*b*).

Both the PDM and SCM domains appear to be similar. However, that is true only in principle, implementations are different. Persson Dahlqvist et al. (2001*a*) have categorized the differences as follows:

- system architecture,
- product model,
- evolution model, and
- process model.

In the following sections issues important in a DES context will be discussed.

5.1.1 System Architecture

Both domains use a server with a database and a server application. The client contains a user interface and some application functionality.

The data representation in PDM and SCM are fundamentally different. PDM uses an object oriented data model where data is represented via a business item connected to a data item. The data representation in SCM is more or less a file system with directories and files. The meta data is stored together with the file itself. In SCM meta data is of little importance while it is the other way around for PDM.

A PDM tool is integrated with various applications and builds an information infrastructure where data from applications is gathered and exchanged. A SCM system can be used as a stand-alone tool, or set of tools. It can also be used as a set of functions that can be used by other tools. SCM tools are often designed to provide information and data to other applications and is easy to encapsulate in other tools. PDM tools provides the central process that creates activities in other tools. PDM has standards defining transfer protocols while SCM uses plain files (Persson Dahlqvist et al. 2001*a*).

5.1.2 Product Model

In PDM an explicit product model is used while there is little support for product models in a SCM system where the product structure is defined in tools such as Make (Feldman 1979). Behind the product model in PDM is a data model, which describes the types of objects, relationships and attributes used in it. Sets of industry specific data models are included in the STEP standard (Persson Dahlqvist et al. 2001*a*).

5.1.3 Evolution Model

PDM recognizes three different concepts for versioning. Historical versioning is conceptual and similar to SCM versioning, dealing with revisions/versions of a product. Logical versioning manages versions of parts as alternatives. Domain versioning is the generation of different views of product structures, e.g. as-planned, as-designed, and as-manufactured.

The emphasis in SCM is on historical versioning, including the possibility to create and merge branches and to present the differences between versions. Logical versioning does not exist. The concept of view is related to the flexibility to create configurations by selection of correct versions of the files included in a specific configuration (Persson Dahlqvist et al. 2001*a*).

5.1.4 Process Model

The process models of SCM and PDM are conceptually the same. A state transition diagram describes, for a product type, the legal succession of states. The alternative way to model processes is the so-called activity centered modeling, in which the activity plays the central role, and the models express the data and control flow between activities (Persson Dahlqvist et al. 2001*a*).

5.2 Software Configuration Management

Technically, the purpose of using SCM is to solve some of the problems related to the evolution of a software product. These problems center around the lack of control and understanding of all the components that make up a product. Furthermore, the product evolution should be coordinated over time by many people (Dart 1990).

The goals of using SCM are to ensure the integrity of a product and to make its evolution more manageable. Integrity is defined as the state or quality of being entire or complete. Bersoff, Henderson & Siegel (1980) defines *product integrity* as being the intrinsic attributes which characterize a product that meets user requirements imposed, assumed, presumed or intended during any stage in its life cycle, which facilitate traceability from product conception through all subsequent stages in its life cycle and which characterize a product that meets specified performance criteria.

SCM can, however, be used more generally to control and manage any set of documents, usually files, for some purpose. The software development community has already faced the size, complexity and concurrent development problems that are facing DES model developers as well. SCM solutions promise to overcome these common development problems (McClanahan 1996):

- lead-time,
- increased size and complexity,

- heterogeneous platforms,
- geographical separation,
- quality-assurance requirements,
- emphasis on reusability and
- impact of programmer turnover.

Although there is overhead involved in using SCM, it is generally agreed that the consequences of not using SCM can lead to many problems and inefficiencies (Babich 1986, Bersoff et al. 1980, Dart 1990).

5.2.1 Software Configuration Management Definition

While there is no single definition of SCM, there are three widely disseminated views from three different sources: IEEE (Institute of Electrical and Electronics Engineers), ISO (The International Organization for Standardization) and the Software Engineering Institute at Carnegie Mellon University. The definition given here is a synthesis of the different views.

Identification An identification scheme is needed to reflect the structure of the product. This involves identifying the structure and kinds of documents, making them unique and accessible in some form by giving each document a name, a version identification and a configuration identification.

Control Control is performed by controlling the release of a product and changes to it throughout the life cycle. This is done by having controls in place to request, evaluate, approve or disapprove and implement changes.

Status accounting Recording and reporting the status of project configuration items and change requests performs status accounting. Such information includes initial approved version, status of change requests, implementation status of approved changes.

Audit and review Audit and review is about validation of the completeness of a product and maintaining consistency among the components by ensuring that components are in an appropriate state throughout the entire life cycle and that the product is a well-defined collection of components.

Dart (1992*b*) adds the following concern to the definition, which is of interest here:

Team work Controlling the work and interactions between multiple developers on a product.

In the definitions, words specific to software development, e.g. code and documentation, could be replaced with the more general word *document* as done in some literature (Asklund & Magnusson 1997, Minör & Magnusson 1993, Olsson 1994, Magnusson, Asklund & Minör 1993). Documents can then be code, models, documentation, CAD models, etc. The more general word document also reflects the more general use of SCM as described in Dart (1992*a*) and Dart (1992*b*) that is applicable to the usage described in this thesis.

5.2.2 The Repository and Sandboxes

Most SCM systems use some kind of notion of a *repository* that is a centralized library of files. Not only the files can be retrieved from the repository but also file history information which includes the different versions of the files, the reason for a change, who replaced that version of the file and when. Usually only the actual difference between each version is stored which reduces space requirements and access time to a particular version. Access time is especially important when files are large and transmitted over a slow network.

To work on a file or collection of files, users withdraw, *check out*, files into their private workspace (sandbox). When done editing, the modified files are *checked in/committed*. A new revision is created of a file when replaced. The private workspace protects the developer from changes made in the repository and protects other developers from changes made locally which results in a stable and controlled work environment for all developers.

5.2.3 Group Awareness

To reduce the amount of communication between developers some sort of group awareness is desired. The larger the project the more important group awareness becomes.

The collaboration mechanism in CVS sends e-mail messages to those who 'watch' a document (Cederqvist 1993). When a developer makes a file editable, locks a file, or commits a file, an e-mail is sent. The e-mails contain information about the performed operation, on what file and by whom. This mechanism reduces the need to communicate who is editing what file thus enabling effective project management. The e-mail mechanism is, however, a rather blunt tool for group awareness.

The *Mjølner orm* project has a better approach and has support for shared revision graphs to view who is editing what (Magnusson et al. 1993, Olsson 1994, Minör & Magnusson 1993). The system also supports both asynchronous and synchronous editing. A synchronous editor allows multiple users to access and edit shared material simultaneously employing a WYSIWIS (What-You-See-Is-What-I-See) metaphor.

Another useful utility is *active diffs* (Olsson 1994). Active diffs are an automatic merge of two revisions of a file that is shown but not saved.

The active diffs and synchronous editing utilities are made for making aware of or showing differences of text files such as program code. These mechanisms are of no use when working with simulation models with obscure code developed through a GUI or binary files such as documents generated by a word processor. The shared revision graphs, however, are usable for all document types.

5.2.4 Configuration Save Mechanism

Two lacks in current SCM systems have been pointed out by Asklund and Magnusson (Asklund & Magnusson 1997). CVS, as well as a number of commercial systems, uses a very simple mechanism for saving configurations. During a certain development stage all current revisions of the documents under control are tagged or labeled. It is then possible to restore the same set of files by retrieving the revisions with a particular tag. However, the configurations in between these tagged events are not known. Opposed to this, systems like Aegis (Miller 1999) and the system in the *Mjølner orm* project (Magnusson et al. 1993), stores the entire configuration each time a file is committed. It should be noted that the possibility to retrieve a set of files from a certain stage in the development process does add considerable functionality compared to no configuration management at all or manual version control.

5.2.5 Managing the System Life-Cycle

A system life cycle can be said to have milestones to mark progress along the path. These milestones are in this context called *baselines*. Bersoff et al. (1980) describes typical baselines in a life cycle:

Functional Baseline The functional baseline ends the system concept formulation stage. Typically this is a requirements document.

Design Baseline The detailed design is critical to successful system development. Generalizing this to any design means that the design is made up of a number of modules that should fit together, i.e. should have well defined physically or logically compatible interfaces. It should be noted that no code or hardware has been built at this stage. The system still only exists as a collection of documents. It may seem as a lot of work is put on producing nothing except a number of documents. However, this approach imposes the efficiency and discipline required to produce a system with integrity.

Product Baseline The product baseline is the first system that works acceptable, but often does not represent the final system implementation. During this development a lot is learned, but there is not time to implement all possible enhancements.

Operational Baseline The operational baseline represents the system used in production. This system includes appropriate improvements learned during the development up to the product baseline.

5.2.6 The Configuration Control Board

The CCB (Change Control Board or Configuration Control Board) has two fundamental responsibilities of configuration control. One is to approve, monitor and control the conversion of *design objects* into system *configuration items*. The other is to approve, monitor and control changes to the system (Bersoff et al. 1980).

Representatives from the developer, customer and user should be members of the CCB. For a simulation project the domain experts can be added to the CCB. The domain experts are important in that they are the suppliers of the information that affects the model design and simulation results.

5.3 Discussion and Conclusions

In the choice in between SCM and PDM one should consider the systems available and the methods used in the organization. In this research the quickest approach was to use SCM and it proved to be sufficient for the usage presented in this thesis. Developing a DES model is quite similar to developing a software product in many ways. One of the main advantages has been the possibility to view changes made in the source code.

In the performed research the usage of SCM has been extended further and all documents created performing the research, including this thesis, has been under SCM control. The benefits of using SCM have been so convincing that SCM will most likely be used in future work as well.

Part III

Case Studies

Chapter 6

The Profilgruppen Case Study

The case study at Profilgruppen in Åseda 1997 was performed to verify a not yet built factory. There were doubts about whether a new planned manufacturing system would cope with the planned throughput. The performance of the two last blocks of the system was considered uncertain due to new equipment. An overview of the new factory is shown in Figure 6.1

The objectives were to find maximal throughput for a specific set of orders and the bottleneck limiting throughput. Another objective was to find ways to enhance the manufacturing system.

The simulation study was performed when the main parts of the new factory was designed. The construction of the factory began before the simulation study was completed.

Input data was a file containing an expected order list with a number of attributes controlling the processing of the parts. No failure processes or other stochastic behaviors were included in the model.

The model conceptualization and translation was performed through interviews. Personnel at the company were educated in basic DES methodology to facilitate the interviews.

Model verification and validation was performed by studying simulation model behavior and by analyzing each code segment together with a domain expert.

6.1 The Studied System

The manufacturing system was divided into three blocks. In the first block the profiles are extruded, cut in appropriate lengths and stacked in a rack. In the second block the racks are stacked and then the profiles are annealed and cooled. The third block consists of destackers, conveyors and packing stations.

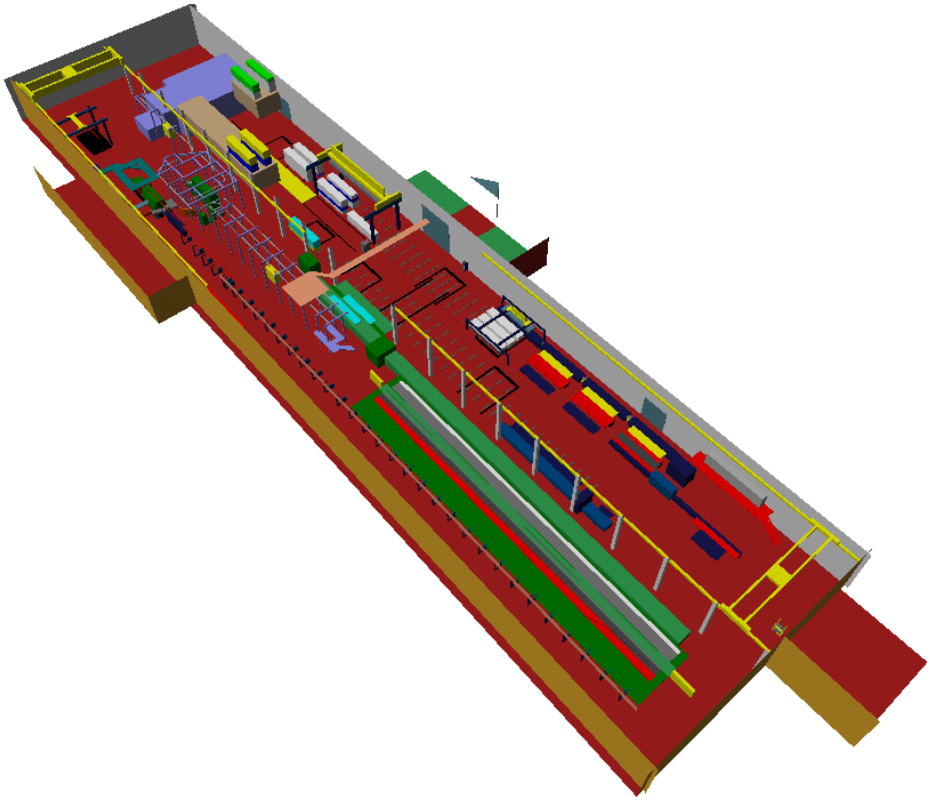


Figure 6.1: *The new factory at Profilgruppen.*

The actions in the second and third block were controlled by controllers that interacted. The controllers in the model simulated the logic in the real controllers and some of the actions performed by operators. The simulated controllers consisted of two parts. The first part received requests and returned signals. The second part did the matching between requests according to a number of priorities and requirements. The controllers and connected elements required two thirds of the 3000 lines of SCL code. Another reason for the vast amount of code was that what was considered being a part was context dependent and thus required coding.

6.2 Results

The simulation showed that orders could be run in certain sequences to reduce idle time. There was a bottleneck at the end of the manufacturing system and most of the time lost was due to transports. These losses could be reduced by changes of the layout. The simulation model was used to analyze the number of racks required in the material flow and it was found that the number of racks could be reduced with 44% without a decrease in performance. As a consequence the WIP (Work In Progress) was reduced as well.

The cost reductions of the proposed changes was substantially greater than the cost of performing the simulation study. A rough estimation, made by the company, of all the possible savings, showed that this project alone could pay for the simulation education, computer, simulation program and the simulation model development without taking into consideration the long term cost reductions.

6.3 Discussion and Conclusions

Some of the proposed enhancements of the manufacturing system could have been found using static calculations. However, the most important enhancements would not have been found without simulation since they were dependent of the dynamic behavior of the system.

One of the reasons not to implement simulation is cost. In this project it was proved that although simulation is costly and difficult, cost can be reduced considerably.

The match mechanism in the controllers was so complex and the logics so intertwined that code maintenance was complicated.

As soon as there is a draft of the manufacturing system a draft model could be made. In this project the simulation was made to late and some of the proposed enhancements was no longer possible to implement.

Chapter 7

The ABB and Volvo Case Study

In 1997 a project was initiated together with ABB BiW (ABB Body-in-White) in Olofström and VCBC in Olofström. The two companies work intimately together when designing new manufacturing systems. ABB BiW designs and delivers complete production lines with VCBC being one of their major customers. Their main interest was to have high accuracy throughput estimations when designing new manufacturing systems.

The case study was developed further in 2002 as a master's thesis work (Pålsson & Quist 2002) supervised by the author and Lars Holst. The focus in the second project was somewhat different, but many of the results from the first case was reused and developed further. The results from the second case study are presented in Chapter 13.

The studied system prepared and married the inner and outer car hood parts in a flexible production line. Robots where used for material handling, processing or to hold and move parts during processing. The production line was divided in three blocks with buffers in between, shown in Figure 7.1.

As can be seen in Figure 7.1, the material flow was straight and simple. However, there was a complex SFCS and robot controllers that complicated model building.

The following main topics could be identified for the project:

- determine DES accuracy for the studied system,
- development of an input data collection method,
- development of an input data analysis method,
- development of an input data generalization method,
- development of an integrated manufacturing system design process

Inaccuracy in predicted manufacturing system performance is costly and even small errors generate high costs over time. To achieve simulation results

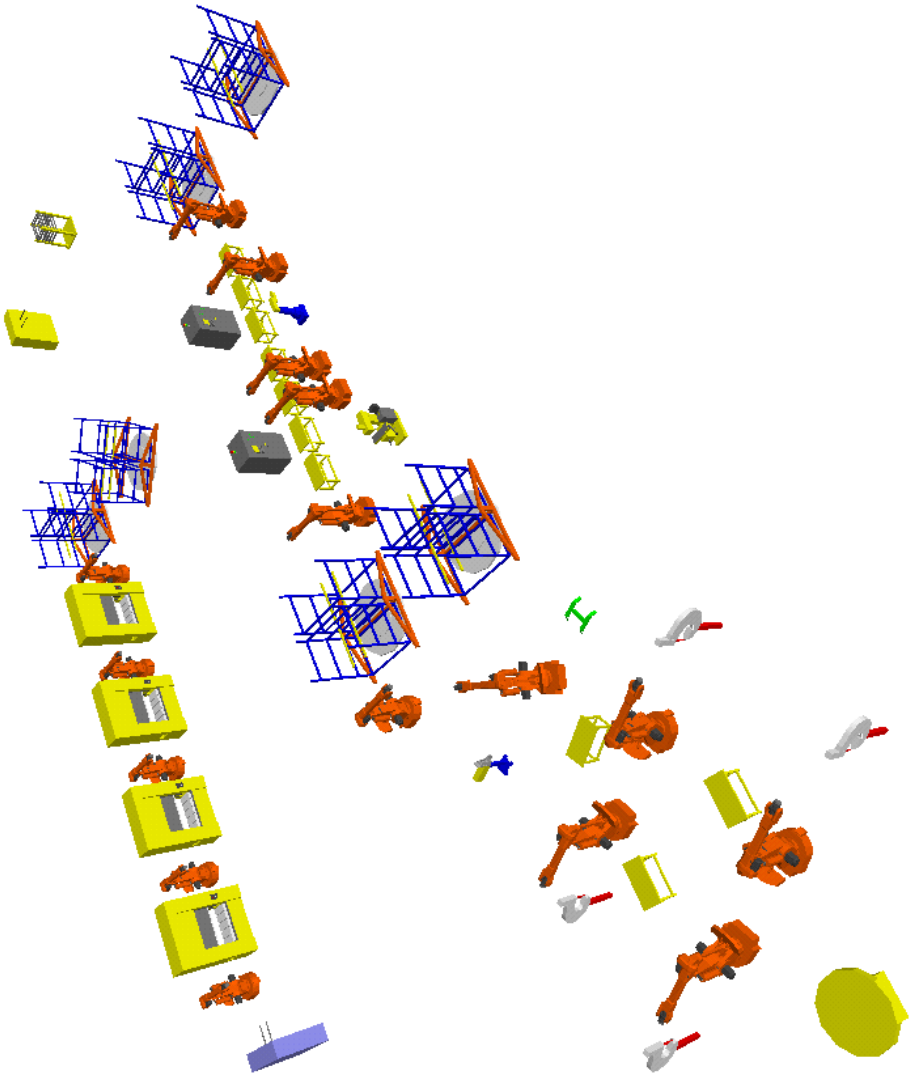


Figure 7.1: *An overview of the system studied at Volvo.*

with high accuracy cycle time estimations, failure distribution estimations, and system logic coding has to be performed with high accuracy.

To validate the accuracy of the throughput measures generated from the DES it was decided that the developed methods should be applied to an existent manufacturing system. The advantage of choosing an existing manufacturing system was that data was available and the model could be compared to the real system. Knowing the accuracy of the modeled system compared to the real system could then be used to assess the accuracy of simulation models in future manufacturing system development projects. Potential errors in the estimations of performance could also be detected and accounted for in future projects.

7.1 Input Data Source

Semiautomatic data collection was installed in the studied manufacturing system and in other similar manufacturing systems at VCBC. Data was used to generate system performance reports. It was to be determined whether the input data available could be used for DES. If usable, a suitable method to transform and analyze data was also required.

A manufacturing system of the studied type can be said to have a hierarchical structure with the levels: line, block, safety zone, cell, device and sensor. A safety zone is the part of the manufacturing system that is stopped when an operator enters the zone, e.g. when performing maintenance.

Operation of the manufacturing system was facilitated by a system referred to as XGOT (GOT (Graphical Operator Terminal)) that worked at the sensor level. The XGOT system displayed the system's state and assisted operators in locating causes of stops.

The XGOT system also delivered data to the data collection system, PANDA, which worked at the cell level. The operators classified stops with stop codes.

There was a 60-second timeout between the XGOT system and the PANDA system, which worked as a low pass filter, i.e. a state with duration shorter than the timeout was not collected by the PANDA system. The extracted production data files had events ordered chronologically with fields for date, time, activity code, activity comment, and shift. Input data was collected for a month.

7.2 Input Data Structure

Production stops was classified as having internal or external causes with respect to the production line, i.e. stops within control of personnel at the production line and those out of control. Stops with external causes, e.g. material starvation, were sampled as being in an inactive (I) state. Down time with internal causes were sampled as being in a stop (S) state. When the cells were operational, but not necessarily operating, they were in the production (P)

Table 7.1: *Selected fields of typical entries from the PANDA database.*

Date	*Time*	*Block*	*Stop Code*	*State*
1998-05-19	15:37:14	190321	nil	P
1998-05-19	15:38:45	190321	nil	S
1998-05-19	15:39:27	190321	nil	P
1998-05-19	16:04:05	190321	6016	S
1998-05-19	16:09:28	190321	nil	P
1998-05-19	17:07:03	190321	5100	I
1998-05-19	17:21:14	190321	nil	P

state. Each stop was identified with a stop code that identified in what cell or block the stop had occurred.

7.3 Stop Codes

The stop codes could be defined for one or several cells and could be overlapping as well, shown in Figure 7.2.

When the required manual classification was not performed the stop code got the value *nil*. In the simulation model the undefined stops were assumed to affect the entire block, which is a reasonable approximation since the entire block will stop when corrective maintenance is performed. On the average only 50% of the stops were classified. The result of this was that the real TBF (Time Between Failures) and TTR (Time To Repair) data samples were known only to some extent.

The stops could be extracted from the database principally in two ways: per cell, i.e. with all stop codes affecting the cell aggregated or per stop code. The latter is preferable since it gives the failure behavior for a certain kind of cell and is thus more generalizable.

Table 7.1 shows a few selected fields from a few typical entries in the PANDA database. To calculate the duration of a state the time for the next state shift was subtracted from the time for the previous state shift. The time in the (I) state was subtracted when calculating TTR and TBF times since the simulation was only run during the operational time.

7.4 Input Data Analysis

Methods for assessment of TBF and TTR distributions was to be developed based on collected production data. Input data analysis methods were to be implemented so that input data analysis could be made automatically or semi-automatically for the studied system and other systems at VCBC.

Filters written in Perl (Practical Extraction and Report Language) were used to convert input data into TBF and TTR times. The filters were highly flexible

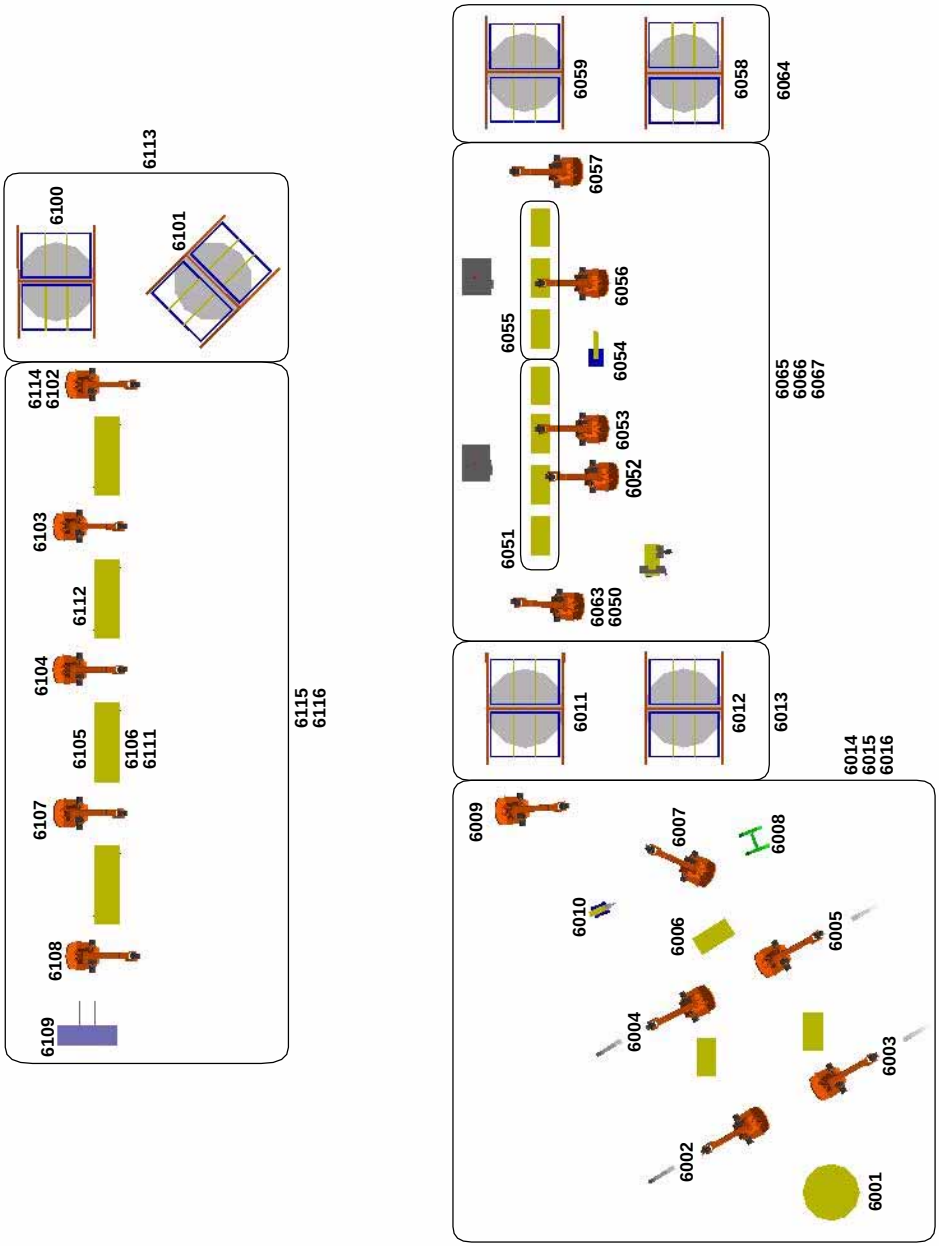


Figure 7.2: *The stop codes in the studied system.*

and could be told to include or exclude one or several stop codes, which made it possible to model the failure processes in several ways. The filters could generate data for file driven simulations, i.e. the time stamps was calculated from a common origin, or for statistical analysis on different time-scales.

A number of routines where written in Matlab for automatic analysis of input data. The routines use conventional chi-square tests with an *equiprobable approach* for goodness-of-fit tests (Banks et al. 2000, Law & Kelton 1991). The routines used the Matlab Statistics Toolbox for parameter estimations and for hypothesis tests. The Matlab Statistics Toolbox parameter estimation routines are based on ML (Maximum Likelihood).

There were 22 data sets which yields 44 failure and repair processes for each model to update. To reduce the time for updating all the failure processes in the models, a BCL script was automatically generated from the output of the Matlab analysis. The BCL scripts were then used to update the simulation models failure and repair processes automatically.

When there were to little data or when data could not be fitted to an analytical distribution file driven failure and repair processes were defined.

7.5 Managing a Family of Models

The simulation study resulted in a family of similar simulation models that was to be managed when performing the simulation study. One of the purposes of SCM is managing a family of products that stems from a common base (Babich 1986) and thus a SCM system was to be tested as a mean to keep track of the different models and their evolution.

Two types of models were built from a base model. The first type of model was built with detailed logic, replicating the SFCS and thus required manual coding of object logic. The second type of model was built mainly through the QUEST GUI and thus reduced the modeling time. The two different types were used to test if it was sufficient with the simpler approach for adequate accuracy. The simple model took 50-70 hours to make while the more complicated one took slightly more than 100 hours. The simplified model type later proved to be an insufficient way of modeling the system and was abandoned.

The two types of models were driven in three ways with respect to failure and repair input data. Data was collected in wall-clock time and could be extracted and transformed to be used in a simulation in different ways:

(a) file driven, i.e. TBF and TTR times were based on raw data read from a file, or driven by distribution functions,

(b) busy-time mode or simulation-time mode or

(c) a mixture of the two approaches.

This resulted in six (2×3) variants of the simulation model derived from a base model, shown in Figure 7.3.

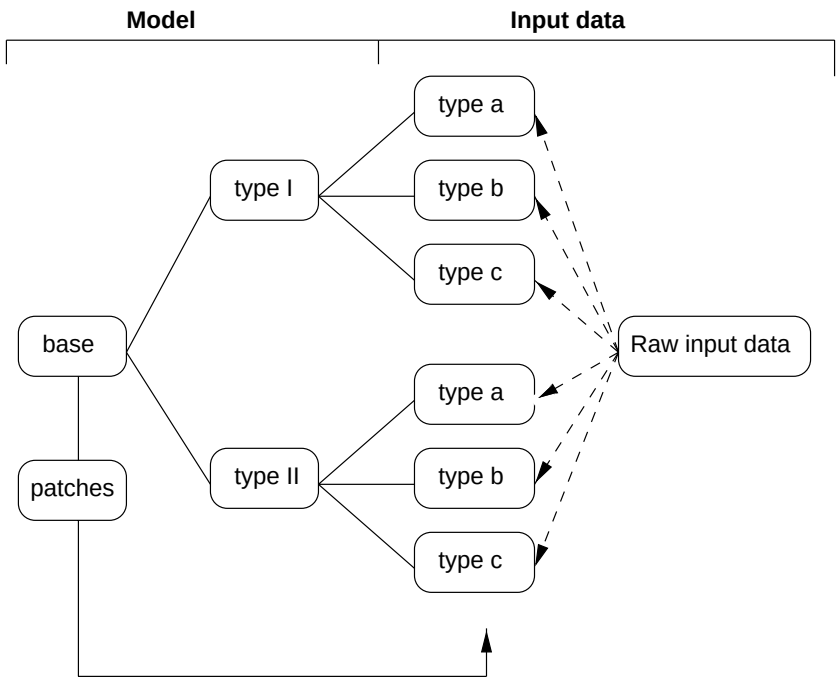


Figure 7.3: *Variant tree of the simulation models with patches merged into head revisions.*

Table 7.2: *Results from the statistical analyzes.*

	Frequency	
Distribution	*TBF*	*TTR*
exponential	3	1
gamma	5	1
lognormal	6	4
Weibull	6	2
Normal	0	0
To little data	12	16
No fit with any of the tested distribution functions	1	5

7.6 Results

Output data was analyzed using conventional confidence intervals. There were not several parameters in the model to be analyzed, omitting the need for advanced methods (described by e.g. Eriksson 1997).

The stops appeared with short intervals and were short in their duration. This kind of behavior is well described by e.g. the lognormal, Weibull, Gamma or exponential distribution. These distributions were therefore used for the distribution function hypotheses. The Normal distribution function was tested as well but could not be fitted to any data. Results from the statistical analyzes are shown in Table 7.2. Note that some data sets could be fitted to more than one distribution function. The chi-square tests were performed with $\alpha = 0.05$.

The statistical analyzes showed that the most appropriate distribution functions were the Weibull distribution and the lognormal distribution. In many cases the distribution hypotheses had to be rejected since there were to few data samples which in turn probably was a result of the large number of unclassified stops.

Building different models from a base model can be done by simply copy a model and modify it. However, by creating branches on a revision tree with a SCM system it is possible to merge changes, or *patches*, in the base model into branches, i.e. the model variants shown in Figure 7.3. If models were copied without using SCM each model variant would have to be modified manually when maintaining the simulation model.

Merging patches into model variants has its application. When performing test runs, a bug inherited from the base model was discovered. The bug could then be fixed by correcting the bug in the base model and merge the patch into the six variants thus reducing the overall maintenance effort for the family of models. This approach works well with source code, but not with binary files and in some instances simulation model files.

7.7 The Scania Data Collection System

As a comparison to the data collection system studied at Volvo another case performed by Ingemansson (2002) is presented. Ingemansson analyzed data from a manufacturing system at Scania in Södertälje. This data collection system collected data with higher granularity.

The data collection system studied by Ingemansson sampled the following states continuously:

- stop,
- warning,
- idle, starved or blocked,
- busy,
- shut off,
- NC program is being executed (sub-state of the busy state)

The data collection system was fully automatic and no manual intervention was required to classify the stops. The stop classes described what subsystem that failed and there were stop classes for manual shutdown and unclassified stops.

One of the problems encountered was that most of the stops where unclassified or the machines where manually shut down. These two classes stood for 65% to 84% of the downtime. In the studied system the planned stops were tool changes which automatically were classified as manual stops. This can to some extent explain the large amount of manual, and thus unclassified, stops.

The unplanned stops had several stop classes. One lack with the data collection system noted by Ingemansson (2002) was that the WT (Waiting Time) was not detected. Even though it is difficult to sample the WT it is desirable to do so since previous case studies have shown that more than 90% of the down time can be WT (Ingemansson & Bolmsjö 2001).

Another lack identified in the data collection system, was the undetected quality losses, i.e. scrap and added work.

The simulation model showed that, with the high quality input data available in this case, it was possible to generate a high accuracy simulation model.

The large amount of unclassified stops should be reduced for usage of the data when improving the manufacturing system. Ingemansson (2002) suggests that a manual classification be added to classify all stops. This classification should then be predefined and simple to enter.

7.8 Discussion and Conclusions

Due to the large amount of unclassified stops it proved to be difficult to validate the model. The model and data filters were modified in several ways to

improve accuracy, but eventually it was concluded that data was not usable for high accuracy simulations. Furthermore, it was revealed during the error tracking process that some of the stop classifications were incorrect. The main reason for not classifying the stops correctly or not at all was probably the required manual entries. It was therefore concluded that data has to be collected automatically.

Mostly it is the complex control systems in the manufacturing system that causes extensive coding. In principle they work the same way, i.e. sending signals and waits for signals before performing an action. The code written to replicate such behavior is usually quite complex and hard to comprehend, especially when maintaining the code. The program code becomes highly unstructured accessing other objects inner details and is thus not at all modular. In the second case performed on the same system the code was modularized and parameterized to avoid this problem, see Chapter 13.

The manual editing of the imported geometries consumed considerable time and is not a practical method if time is an issue. As a result most of the available geometries were not used in the simulation model. However, other methods have not been found and this method seems to be common practice. It should be noted that direct import from the original CAD format or STEP format and more powerful computers would reduce this problem in the near future.

Chapter 8

The First BT Products Case Study

The case study described here was performed in cooperation with BT (BT Products), a world leading manufacturer of electrical warehouse trucks in Sweden. There were four simulation model developers at two geographically separated sites. The developers worked in a heterogeneous environment using MS Windows NT and IRIX respectively.

A manufacturing system for truck mast manufacturing was studied, shown in Figure 8.1. The simulation study was explorative and performed to find possible ways of reducing lead-time.

The research objective was to develop and test a methodology for concurrent development of DES models in a heterogeneous environment. The company's objective was to reduce simulated lead-time by 25% and the throughput was to be increased with 25%.

8.1 Configuration Management

Each developer was assigned modules to develop. A structured naming convention was used to facilitate a later merge of the modules.

Model files were locked in the SCM system to avoid merge conflicts while SCL sources were kept unlocked. The developer that locked a file owned it until it was unlocked or checked in to the repository.

Concurrent development was performed through a central repository via Internet. The repository was set up on a SGI (Silicon Graphics) IRIX computer in Lund. The SCM system kept track of all the file revisions, who edited a file and comments regarding what changes had been made.

Group awareness was facilitated through the watch mechanism in CVS that was set up to send an e-mail message every time a developer was making a file editable, locked a file, or committed a file. The e-mails contained information about what operation had been performed, on what file and by whom. Such

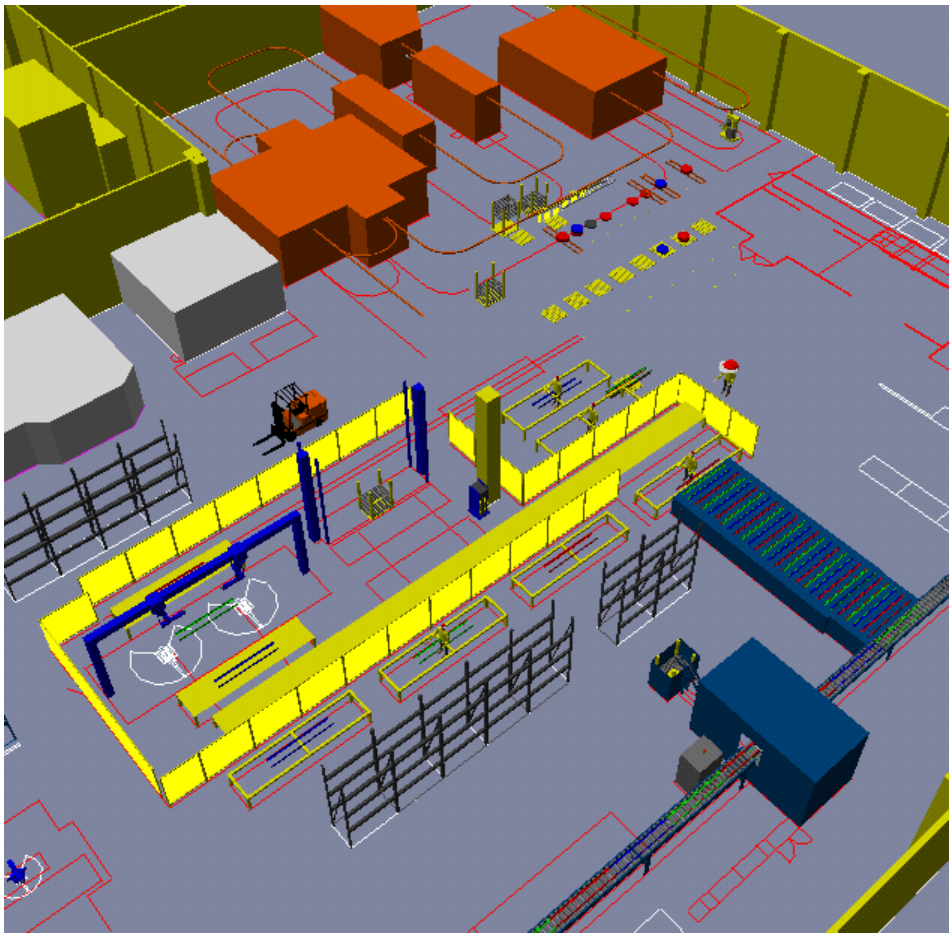


Figure 8.1: *The central part of the simulation model at BT.*

a mechanism reduces the need to communicate who is editing what file and thus facilitate project management.

Instead of a formal CCB, change requests was documented in the meeting protocols containing information about what had been performed, what was to be performed and by whom.

8.2 The Model

For effective concurrent development a draft base model with a detailed layout was split into sub-models, or modules. When all the modules were ready they were merged again to form the final model.

During development of the modules *stubs* were used, replacing the not yet developed modules connected to the module under development. The stubs did not have logic but generated the same kind of output or accepted the same input with the same interface as the real module would have done.

The model consisted of an NC-machine, an automated robot cell, a paint shop and a number of assembly stations. The studied system had several interactions with the surrounding system, which complicated input data analyzes.

The studied systems logic was mapped through interviews with planners, the production engineer and operators. The conceptual model was created and documented as the project went along. Data was collected from the scheduling system, a SFCS and manually with a stopwatch when not available in digital format. Some of the data with long cycle times had to be collected for complete days, and still only a few samples was obtained.

For each module there was a file for each type of logic, e.g. route, request, process, etc., named with a structured naming convention. The structured naming convention made it easy to keep track of the sources and avoided name clashes when later merging the modules. A separate directory contained include files for common definitions and declarations, such as constants, attributes and routines.

To a large extent the failure, repair and cycle processes were file driven. Either there was to few samples to perform a chi-square goodness-of-fit test or no analytical statistical distribution could be fitted.

Approximately 2800 lines of code were written in this project to add functionality to the model that was not achievable through the GUI. As in the other case studies described here, it was complex control systems that caused much of the coding together with manufacturing control rules.

8.3 Communication Channels

Large simulation models are inherently complex. With multiple developers project management becomes even harder due to the increased need for communication, as demonstrated in Figure 8.2 (Babich 1986).

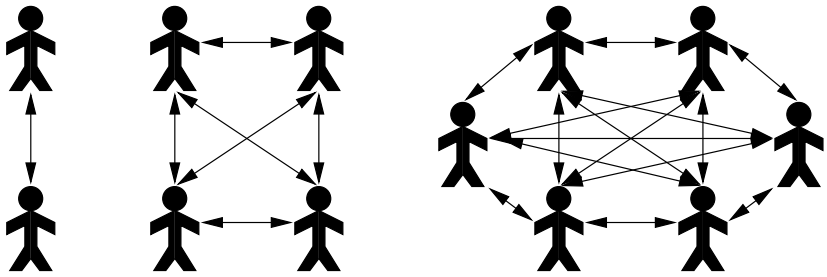


Figure 8.2: *The increasing number of communication channels with multiple developers. In this case two, four and six developers. Adopted from Babich (1986).*

The case was used for training at BT and thus extra communication was required which slowed down the development process. Developers had more knowledge about each other's modules than required.

8.4 Documentation

The documentation was hard to keep updated as development progressed and new information became available. Documentation was transferred via e-mail in several instances, causing the expected problems with different versions on different computers.

Documentation of the SCL code was fairly simple since the user is in control and can add as many comments as needed. However, when it came to extracting the source documentation into the project report there was a vast amount of *cut-and-paste* needed. Once that was done the maintenance of the model documentation force the modeler to update the documentation in the model source as well as in the model documentation. This is what Babich (Babich 1986) refers to as the *double maintenance problem*. A solution to this problem was developed after this project and is presented in Chapter 11.

8.5 Results

The first bottleneck was identified, and by removing that bottleneck the next bottleneck was found and so on. Thus the maximum capacities for each of the elements in the model was identified as an basis for an investment plan. That is, if an investment was planned to remove a bottleneck it was known in advance where the next bottleneck would occur and what the maximum throughput for the entire system would be. This is a useful result when comparing different investment alternatives and when performing the investment analysis.

The desired reduction in manufacturing lead-time could not be verified since there was no data available.

One of the main experiences in this project was that a simulation study should be performed continuously, i.e. without to long interrupts. The interrupts in this case generated much overhead since the developers had to get into the conceptual model repeatedly before they were able to do any productive work.

8.6 Discussion and Conclusions

It was experienced that the need for structure increases drastically with the number of developers. Structure is needed in all activities of a simulation study, i.e. there has to be a well defined structure of:

- the organization,
- the project plan,
- the method,
- the documentation and
- the simulation model.

As with CM, formal structure generates overhead. However, the overhead is small compared to the increase in process efficiency. Without a well defined structure the informal communication increases although being hard to detect. Documentation of the structure and documents describing the project details was an important tool to reduce this communication. The documentation was also an important part of the verification and validation process.

The problems that arose with filenames and revisions when files where exchanged by other means than through the SCM repository showed that multiple developer projects have to use some notion of a central repository and a mechanism for locking files or allowing concurrent editing of files.

CVS made it possible for the developers to work with files locally without disturbing the work of others and collaborate over distance transparently. The need to lock certain files did not pose a problem since the model was divided into modules.

The difficulty to keep the documentation updated showed that a different approach had to be implemented. Generating the main part of the documentation before generation of the model would provide a better base for the modeling effort and allow verification of the conceptual model before implementation of the simulation model.

Documenting the simulation study is a complex problem when the simulation software has as structure like the one in QUEST. It is desired that the documentation is in one place to avoid the double maintenance problem and at the same time it should be easily available for the part of code, or element under study at the moment.

Chapter 9

The Second BT Products Case Study

The BT factory in Mjölby was increasing their production volumes rapidly. To accomplish this, investments were made in equipment to store and process the beams for the truck mast manufacturing system described in Chapter 8.

This case study differed from the previous in that it was performed for verification of a new system while the former was performed to improve an existent manufacturing system.

The studied system had been designed and the simulation was to verify that the system would be capable of the specified capacity. The analysis of the simulation was to find:

- bottlenecks,
- buffer placement and size,
- balancing the material flow,
- maximum capacity if the new plasma cutter was not operable, and
- impact of different product sequences

As in the previous case study, the project team consisted of personnel from both Lund University and BT. From BT there were both manufacturing system engineers and simulation engineers.

9.1 System Description

A number of attributes controlling cycle times and routings attached to the parts was collected. The attributes were then stored in an order list. The information content in this order list differed from the information content of the order list in the former case study.

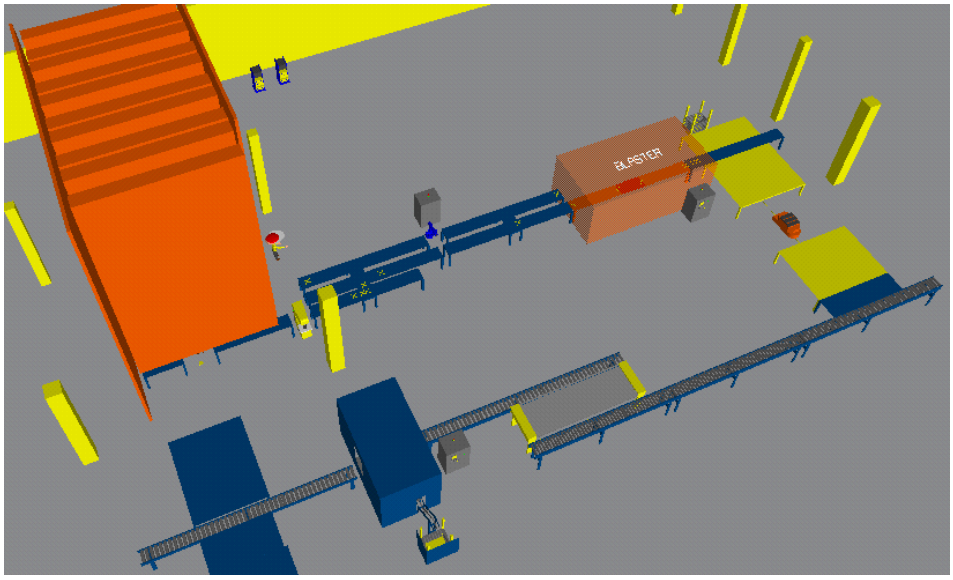


Figure 9.1: *The new truck mast system simulation model at BT.*

The system started where the beams for the truck masts were unloaded from the trucks. The beams were then loaded into an automatic storage. The beams were then retrieved for processing according to an order list at the other end of the storage. The beams were cut and sent to a grit blast machine, in some cases via a plasma cutter. The beams were then transported to the CNC (Computer Numerical Control) mill that was the first element of the simulation model described in Chapter 8. The studied system is shown in Figure 9.1.

Since this was a new system there was no TBF and TTR input data available. Most TBF distributions were estimated by domain experts to be exponentially distributed while the repair distributions were estimated with maximum and minimum TTR times.

9.2 Verification and Validation

Being a new system the validation of the simulation was difficult. In order to verify and validate the model, verification of input data, running and observing model behavior, and verification and validation of simulation results was performed. Each activity was performed together with domain experts at BT. The verification and validation was performed on a base model that was modified for different scenarios.

9.3 Experiments

There were nine different cases that were run to study the proposed manufacturing system. Loading the beam storage system proved to have a large impact on performance and therefore loading at different times of the day was tested. A different configuration of the storage system was also tested. One of the models tested how the system would perform without the plasma cutter and it was tested how improved cycle times in the CNC mill would affect performance. These different changes of the base model were then combined to generate the nine scenarios.

9.4 Results

The simulation results identified one major bottleneck and a second bottleneck if the first bottleneck was removed. The systems functionality and performance was verified. The production sequence was found to have little impact on system performance, but had an impact on buffer sizes. The conclusion was that the buffers were sufficient, but could be reduced with a better sequence. Work procedures could be developed from the simulation model to enhance performance by performing beam storage loading in a certain pattern. The studied system is now implemented and running.

9.5 Discussion and Conclusions

Experiences from the previous case study were implemented in this case study. In the first case study it was concluded that continuity was important and therefore the project was performed at a high pace.

It was concluded in the first case study that SCM facilitated project management, and it was therefore decided to use SCM in this case study as well. However, a central repository could not be implemented in this case study due to technical problems. SCM was therefore implemented by exchanging files via email and performing the required configuration management on a computer in Lund. Being aware of the problem with multiple developers, files were *locked* via verbal communication, i.e. one developer explicitly told the others that he owned the file until further notice. Through these manual routines in combination with the use of the SCM system the documents in the project was kept under control for the duration of the project.

The methodology presented in Chapter 10 was implemented to a large extent. Functional, conceptual and design documents were prepared before model building began. The documents were expanded as work proceeded.

As expected, many possible problems were detected during the first phases of the simulation study and additional information was therefore gathered to reduce the risk of the project.

Part IV

Contributions

Chapter 10

Discrete-Event Simulation Methodology

DES is well covered in the literature with a number of textbooks and papers covering several areas associated with the technique. This chapter presents a methodology for performing simulation study that is also well covered in the literature. However, presented here is a methodology with an emphasis on the management of the simulation study and the integration of the simulation process in the manufacturing system development process and thus the product realization process.

According to a survey on development, operation and maintenance of manufacturing systems in Swedish companies, all companies use some kind of project management strategy, which give guidelines for how the project should be carried out. However, these guidelines are generally not documented and do therefore not provide information on how tasks should be performed. The working procedure mainly relies on experiences from past projects (Gullander & Klingstam 1998). The presented methodology is an attempt to provide a base for a working procedure, based on experiences from case studies, for DES studies in the context of manufacturing system analysis and development.

10.1 Capability Maturity Model

The CMM (Capability Maturity Model) for software, adopted from (Paulk, Curtis, Chrissis & Weber 1993*a*), provides a model for how organization can improve their DES maturity and describes appropriate future research.

Paulk et al. (1993*a*) begins with a statement, in a software development context, that applies equally well in a DES context.

> After two decades of unfulfilled promises about productivity and quality gains from applying new software methodologies and tech-

> nologies, industry and government organizations are realizing that their fundamental problem is the inability to manage the software process ...
>
> Success that rests solely on the availability of specific individuals provides no basis for long-term productivity and quality improvement throughout an organization. Continuous improvement can occur only through focused and sustained effort towards building a process infrastructure of effective software engineering and management practices.

Most DES professional would agree on this quote if the word 'software' was exchanged with the word 'simulation'.

10.2 Process Maturity Levels

The CMM describes the levels of process maturity. *Process maturity* is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. By institutionalizing the process via polices, standards, and organizational structures, an organization gains in process maturity. *Institutionalization* entails building an infrastructure and a culture that supports the methods, practices, and procedures so that the organization endures in case of personnel turnover (Paulk et al. 1993*a*).

As stated by Paulk (1998), obtaining senior management sponsorship is a crucial component of building organizational capability. Individuals can exercise professionalism and discipline within their sphere of control. If an organization as a whole is to change its performance senior managers must actively support changes.

According to Paulk et al. the five maturity levels are characterized by, see Figure 10.1:

(1) Initial The process is characterized as ad hoc. Few processes are defined, and success depends on individual effort.

(2) Repeatable A basic project *management process* is established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes.

(3) Defined Both the *management* and *engineering processes* are documented, standardized, and integrated into a standard process for the organization.

(4) Managed Detailed measures of the process and product quality are collected. Both the process and products are *quantitatively understood* and controlled.

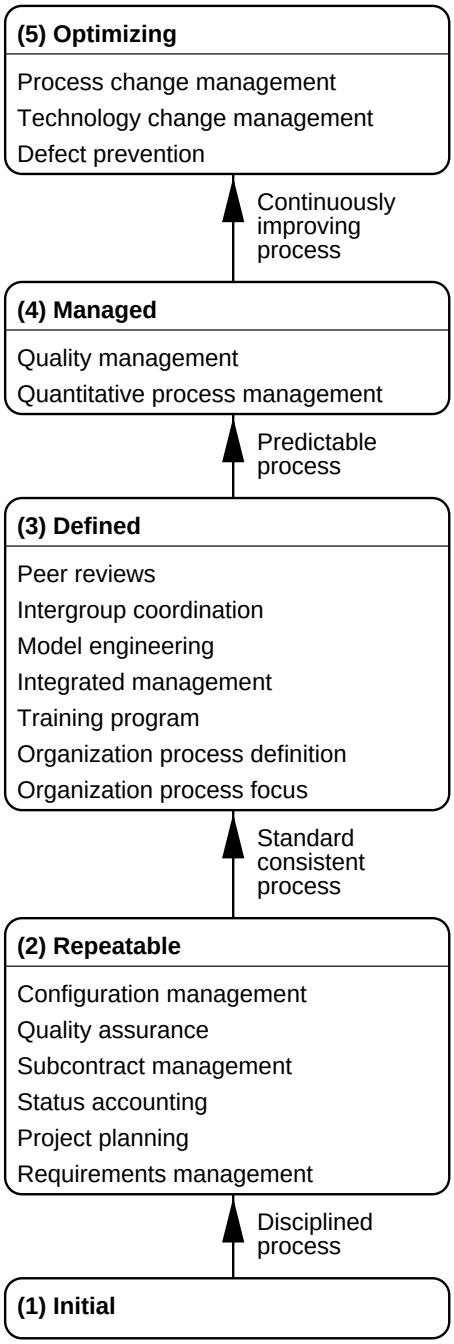


Figure 10.1: *The five levels of process maturity with key process areas. Adopted from Paulk et al. (1993a).*

(5) Optimizing *Continuous process improvement* is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

The presented work aims at providing some of the material required reaching Level 2 and to some extent Level 3. This might seem like a less ambitious objective, but it should be related to common practice where Level 2 hardly has even been considered. According to Paulk et al. it can take several years to move from Level 1 to Level 2, and moving between the other levels will usually take on the order of two years, all in a software context. This is thus not an effortless task, but is considered necessary to make the DES process mature and reliable. DES organizations are in general smaller than their software counterpart, but that does not imply that capabilities are less important.

10.3 Key Process Areas

Key process areas identify the issues that must be addressed to achieve a maturity level, see Figure 10.1. Each key process area identifies a cluster of related activities. To achieve a key process area all the goals of that area has to be performed collectively. When the goals of a key process area are accomplished on a continuing basis across projects, the organization can be said to have institutionalized the process capability characterized by the key process area (Paulk et al. 1993*a*).

The key process areas are categorized in Table 10.1 (Paulk, Weber, Garcia, Chrissis & Bush 1993*b*). The management process category contains the project management activities. The organizational process category contains the cross-project responsibilities. The engineering process category contains the technical activities.

Each key process area is described by key practices that describe the infrastructure and activities that contribute to the implementation and institutionalization of the key process area (Paulk et al. 1993*b*).

10.3.1 Level 2 Key Process Areas

The key process areas at Level 2 concerns establishing basic project management controls. Descriptions of each of the key process areas for Level 2 are given in the following sections.

Requirements Management

The purpose of requirements management is to establish a common understanding of the requirements that will be addressed in a project. This agreement with the customer is the basis for planning and managing the project.

- Requirements allocated are controlled to establish a baseline for model development and management use.

Table 10.1: *Key process areas assigned to process categories. Adopted from Paulk et al. (1993b).*

Levels	*Management*	*Organizational*	*Engineering*
(5) Optimizing		Technology change management Process change management	Defect prevention
(4) Managed	Quantitative process management		Quality management
(3) Defined	Integrated management Intergroup coordination	Organization process focus Organization process definition Training program	Simulation engineering Peer reviews
(2) Repeatable	Requirements management Project planning Project tracking Subcontract management Quality assurance Configuration management		
(1) Initial	Ad hoc processes		

- Plans, project team, and activities are kept consistent with the requirements allocated.
- The parties agree on the requirements.

Project planning

The purpose of project planning is to establish reasonable plans for performing and managing the project.

- Estimates are documented for use in planning and tracking the project.
- Project activities and commitments are planned and documented.
- The parties agree to their commitments related to the project.

Status Accounting

The purpose of status accounting is to establish visibility into actual progress so that management can take effective actions.

- Actual results and performances are tracked against the plans.
- Corrective actions are taken and managed when results and performance deviate significantly from the plans.
- Changes to commitments are agreed to by the parties.

Subcontract Management

The purpose of subcontract management is to select qualified sub-contractors and manage them effectively. It combines the concerns of requirements management, project planning, and status accounting for basic management control, along with necessary coordination of quality assurance and CM.

This issue especially concerns organizations that use sub-contractors to develop and implement the manufacturing system and where the DES model is a part of the documents supplied with the system. This situation has been noticed to emerge lately. Subcontract management is also an issue when consultants are building the model or parts of it.

- The customer and the sub-contractor agree to their commitments to each other and the commitments are documented.
- The customer and the sub-contractor maintain ongoing communications.
- The customer tracks the sub-contractor's results and performance.

Quality Assurance

The purpose of quality assurance is to provide management with appropriate visibility into the process being used by the project and the documents created. This is vital if models are to be reused and if sub-contractors builds the model or parts of it. Note that this is the quality assurance of the process and how the model is built, which is not the same as verification and validation of the model.

- Quality assurance activities are planned.
- Adherence of model and activities to the applicable standards, procedures, and requirements is verified.
- The parties are informed of quality assurance activities and results.

The base for the audits is the documents generated so far. Easily browsable documentation, e.g. HTML (HyperText Markup Language) or PDF (Portable Document Format) documents placed on the intranet, facilitates such activities. The use of an SCM system facilitates the audit process by supplying a *change log* used to review what documents has been changed, in what way, by whom and what new documents have been added.

Configuration Management

The purpose of configuration is to establish and maintain the integrity of the products of the project throughout the project's life cycle.

- CM activities are planned.
- The CM organization is defined.
- Selected tools, methods, procedures and reusable modules are identified, controlled, and made available.
- Changes to identified tools, methods, procedures and modules are controlled.
- The parties are informed of the status and content of baselines.
- CM activities are performed continuously.

Several simulation studies have been performed or supervised and one of the experiences is that people in general have difficulties keeping track of the documents pertaining the simulation study. There is a lack of structure, naming conventions, version handling and CM. Time is thus spent cleaning up the mess created and restoring lost information.

Utilizing an SCM system facilitates the task of keeping track of documents. All documents in a project are related to each other in one way or another and should therefore be under CM control. The methodology strives to keep all

project documents under control for the entire life cycle of the simulation model. CM is thus a vital part of the methodology. As stated by Paulk (1998), a small project might not require a CM group or a CCB, but CM and change control are always necessary.

The concept that all documents in a project are part of a configuration can be extended further. Most documents are generated using some tool and different versions of a tool might not be compatible. To avoid future problems, notes on the working environment, simulator version, etc., is documented. In short, anything that affects the simulation study should be under CM control.

10.3.2 Level 3 Key Process Areas

The key process areas at Level 3 address both project and organizational issues, as the organization establishes an infrastructure that institutionalizes effective DES processes and management processes across all projects. Descriptions of each of the key process areas for Level 3 are given in the following sections.

Organization Process Focus

The purpose of organization process focus is to establish the organizational responsibility for process activities that improve the organization's overall process capability. The result of the organization process focus activities is a set of process assets that are used by the projects.

- Process development and improvement activities are coordinated across the organization.
- The strengths and weaknesses of the processes used are identified relative to a process standard.
- Organization-level process development and improvement activities are planned.

Organization Process Definition

The purpose of organization process definition is to develop and maintain a usable set of process assets that improve process performance across the projects. These assets provide a stable foundation that can be institutionalized via mechanisms such as training.

- A standard DES process is developed and maintained.
- Information related to the use of the standard process by the projects is collected, reviewed, and made available.

Training Program

The purpose of training program is to develop the skills and knowledge of individuals so they can perform their roles effectively and efficiently. The organization is responsible for training, but the project should identify required skills and initialize training when necessary.

- Training activities are planned.
- Training for developing the skills and knowledge required to perform management and technical roles is provided and performed.

Integrated Management

The purpose of integrated management is to integrate the engineering and management activities into a coherent, defined process for a specific project, which is tailored from the organization's standard process and related process assets. Integrated management evolves from project planning and audit and review at Level 2.

- The project's defined process is a tailored version of the organization's standard process.
- The project is planned and managed according to the project's defined process.

Simulation Engineering

The purpose of simulation engineering is to consistently perform a well-defined modeling process that integrates all the modeling activities to produce correct models effectively and efficiently, see Section 10.10.

- The activities are defined, integrated, and consistently performed to produce the model.
- Reusable modules are kept consistent with each other.

Intergroup Coordination

The DES process should be integrated, coordinated and controlled with other groups, e.g. manufacturing process engineers, SFCS programmers, etc. The purpose of intergroup coordination is to establish a means for the simulation engineering group to participate actively with the other engineering groups so the project is better able to satisfy the customer's needs. Intergroup coordination extends beyond simulation engineering.

- The customer's requirements are agreed to by all affected groups.

- The commitments between the engineering groups are agreed to by the affected groups.
- The engineering groups identify, track, and resolve intergroup issues.

Peer Reviews

The purpose of peer reviews is to remove defects from the model early. An important corollary effect is to develop a better understanding of the model and of the defects that can be prevented. Peer reviews have proven highly effective in detecting errors prior to test. Reductions in coding errors of as much as 80 percent have been shown (Conwell et al. 2000).

- Peer review activities are planned.
- Defects are identified and removed.

10.4 Simulation Study Participant Roles

The participants in a simulation study are here dubbed customer, domain experts, and simulation engineers. For simplicity, the customer and user roles have not been separated and the project manager is implicitly understood to exist. The manager's responsibilities have been discussed in the previous sections.

The customer is responsible for formulating the objectives and requirements of the simulation study. The simulation engineers and domain experts aid in this process to avoid objectives not realizable.

Since a simulation study usually cross traditional domain borders the domain experts come from several domains, e.g. production engineers, managers, operators, etc. The domain experts are the suppliers of the information required to build, verify and validate the simulation model.

The simulation engineers compile the information to generate the simulation model. The simulation engineers also build, verify, validate, and execute the simulation model. The simulation engineers work tightly together with the domain experts to create the conceptual model and when verifying and validating the simulation model.

10.5 Simulation Model Properties

This section presents desired model properties that can be used as a checklist in the functional, conceptual and design phases.

Several considerations can be made to enhance the model's usability over time. The model and it's components should be *general*, i.e. the range of utilization should be as large as possible. The simulation model should, if possible, be *domain independent*, e.g. a model can be used both for testing the

control logic in a system as well as assessing the performance measures. A life cycle analysis is made to evaluate what the simulation model will be used for. However, the scope should be as narrow as possible to reduce complexity.

When appropriate, the simulation model should be made *extensible*. For instance, commonly a part of a manufacturing system is studied and it might be of interest to extend the model later to include other parts, as described in the BT case studies. Thus, when working with models incrementally or modular models, *consistency* is vital.

A model should be *efficient*, i.e. the model's ability to efficiently support problem solving and reasoning without the need for any transformation. The model should be *perspicuous*, i.e. easily understood by the participants in the simulation study and especially those affected by the simulation study. Here the graphical presentation of the model is one of the important properties (Savén 1995) and then of course the documentation. It is to be considered what it is worth spending hours on graphic appearance. It is also to be considered how output data should be collected, analyzed and presented.

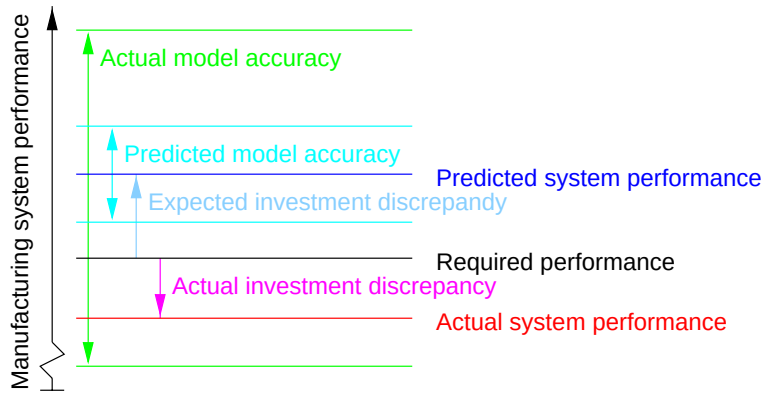
A model should be *complete* in that it contain all the information necessary to solve problems that it is supposed to be able to solve (Vernadat 1996). The simulation is also supposed to be *accurate*, i.e. yield accurate assessments of performance measures and mimic the systems dynamic behavior accurately. It is to be considered what the required accuracy is since there is no point in investing in productive but expensive production equipment. Nor is there any point in investing in cheap equipment that will reduce the manufacturing system's overall performance. The simulation model accuracy will have implications in the risk analysis of the investment, shown in Figure 10.2. It should be noted that simulation accuracy is hard or even impossible to determine if audits and reviews of previous cases have not been performed.

10.6 Simulation Study Properties

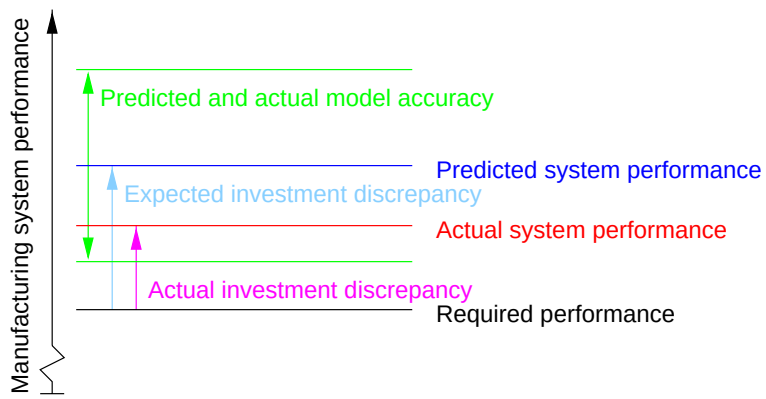
A simulation study should be *complete*, i.e. answer the questions posed in the functional phase.

A simulation study should be fully *documented*. This includes not only the model, but also what the model is based on, methods, etc. All documents pertaining the simulation study should be *maintainable* if the model is to be used over time.

The former properties are closely related to the *credibility*, i.e. the simulation study participants and those affected by the simulation study should believe in the results which is vital for successful implementation. It is believed that active participation, and thereby verification and validation of the simulation study, is a vital prerequisite for credibility. When the participants in the simulation study actively have supplied and reviewed the information that constitutes the simulation study they are bound to believe the results assuming that they have confidence in the technology behind the results.



(a) *Undesired accuracy and risk management.*



(b) *Desired accuracy and risk management.*

Figure 10.2: *Model accuracy effects on the risk of the investments studied. The model accuracy is a function of logic accuracy and input data accuracy. Note that the actual system performance is within the actual model accuracy limits in both cases. Similarly, the required performance is below the lower predicted accuracy limits in both cases.*

A simulation study should be delivered *on time* at a *low cost*. The cost of a simulation study does not only consist of the time spent, but also includes the infrastructure required to perform a simulation study, e.g. simulator, PDM system, translators, training, input data collection systems, etc.

10.7 Simulation Methodology Properties

A simulation study methodology should be *reusable*, i.e. should be transferable from one project to another (Holst, Randell & Bolmsjö 2000*a*). A simulation study methodology should be *life cycle supportive*, i.e. support all phases in the simulation study which may include all phases from the functional phase of the system to the final removal of that system. It may also be a limited study that is used for a limited time of the studied systems life cycle. A simulation study should reduce *nominal time consumption* and *lead-time consumption* for the manufacturing system development process. This can not always be achieved, but it is argued that the quality of a simulation study results are higher than if simpler methods are used and thus achieving a price/performance advantage. As a consequence of the lead-time requirement, a simulation study methodology should support *concurrency*. Furthermore, a simulation study methodology should allow performance in *distributed* and *heterogeneous* environments.

10.8 Documentation

The proposed methodology is based on continuous and incremental generation of documents where the documentation is a vital part. Each baseline concluding one phase is the base for the next phase. Accordingly there are no separate phases for documentation and reporting.

Here the term *document* is used in a general way, i.e. a document can be a project plan, input data file, model file etc. A document is thus not the same as documentation that is a document describing a part of a document or one or several other documents.

Documentation is time consuming, but necessary during the life cycle of a simulation study. It is believed that, with a life cycle view, the net time is reduced when proper documentation procedures are followed. Documentation lives over time as opposed to verbal communication.

- The documentation communicates the current state of the project among the project team members and spreads the accumulated knowledge in the organization.
- The documentation communicates results and findings.
- Documenting general or specific experiences facilitates future projects.

- The documentation ensures model quality and credibility by being open for discussion to all project members. Documentation is also required in the verification and validation process.
- Documentation is a protection against large time and knowledge losses in case of personnel turnover.
- Maintaining a simulation model without documentation is difficult. Documentation of the model and all things that affect the model, e.g. input data, requirements etc., facilitate maintenance and CM ensures that all documents are traceable.

10.9 Verification and Validation

As pointed out by several authors, verification and validation should not be viewed as separate phases, but as continuous actions carried out throughout the project. Therefore there are no separate phases for verification and validation in the process described in Section 10.10.

By continuously documenting and evaluating the simulation project with increased granularity, the model is to a large extent documented, verified and validated when finished.

10.10 Simulation Study Phases

The phases presented here are similar to the phases presented by e.g. Banks et al. (1996), Law & Kelton (1991) and Harrell & Tumay (1995). A few phases have been added and some activities are performed continuously. The proposed DES process is presented with IDEFØ notation in Appendix A and is commented in more detail in the following sections.

The simulation study have been divided in the following phases:

1. functional phase,
2. conceptual phase,
3. design phase,
4. realization phase,
5. experimental phase,
6. implementation phase, and
7. review and learning phase.

I want to make clear the difference between what is referred to as an iterative modeling process by e.g. Pidd (1996) and Harrell & Tumay (1995) and what is here referred to as an iterative modeling process. It is here stated that the simulation study should not be an iterative process, instead it should follow the phases presented here to the extent possible. However, in the conceptual and design phases the work performed is very much iterative as described in Willemain (1994) and Willemain (1995). It is also recommended to look ahead to assess the implications of a decision in the later phases. However, the simulation model should not be built iteratively and incrementally. The argument is that what takes a few minutes or even seconds in a human brain, takes hours to implement in a simulation model. Thus, if the model building would be performed iteratively it would take hours to rebuild the model for each iteration.

Although presented here as a linear development one realizes that changes of the baselines might be needed. This results in change requests and then in updates of the baselines. However, the whole point of the methodology is to reduce the number of back-steps by investigating everything thoroughly in advance. By formally issuing change requests, their refusal or acceptance and implementation is under control. If a change request is accepted for implementation the activities proposed here are performed for the change. That is, each change has to go through all activities necessary up to the current point in the process. This might look like overdoing things. However, if one of the relations in between requirements and how to solve that requirement is neglected serious problems might arise in the realization phase or the experimental phase. It is here implicitly understood that the CCB consists of at least one representative from each of the parties in the simulation study team.

An important aspect of the proposed methodology is that the documents of the simulation study are created and updated continuously. That is, the study begins with problem identification and ends with a complete set of documents. For each phase documents are added, extended or edited. Each phase ends with a so-called *base line*. A base line is a verified set of documents that constitutes the requirements and the information required for the next phase.

In the audits it is considered whether the simulation study should be disrupted, aborted or continued, based on the current situation and the information collected so far in the study. Often enough information is revealed during the early phases to base a decision on. For instance can simple assessments reveal a major bottle neck and possible solutions. The study can then be disrupted until the effect of the implemented enhancements have been evaluated, or canceled if there seem to be no point in continuing the study. The study can also continue, but with other methods than DES, if it is revealed that more feasible tools can be used to achieve the objectives of the study, see Figure 10.3. The latter approach can still be performed within the same methodological framework.

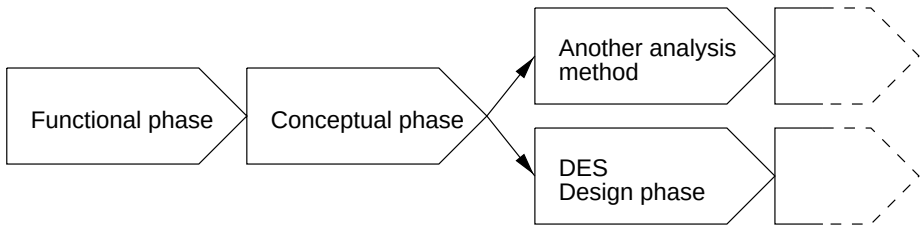


Figure 10.3: *Manufacturing system development process with or without simulation.*

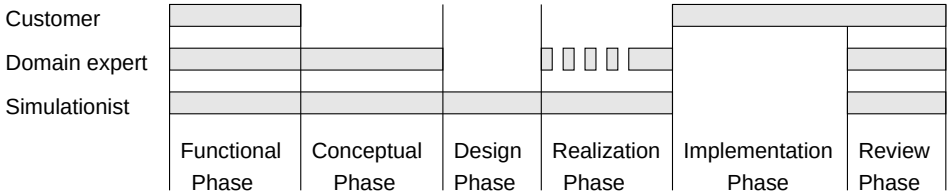


Figure 10.4: *Resource allocation in the simulation study.*

10.10.1 The Functional Phase

The *functional phase* is typically a requirements document. There should be consensus on the requirements between the simulation engineer, domain experts and customer. The responsibilities of the customer and domain experts are not restricted to this phase, but several phases, shown in Figure 10.4.

When the simulation study is a subprocess of the manufacturing system development process, the base requirements document is produced in that process, see Figure 10.5. The integrated simulation study can then be developed according to the incremental concept described by Randell et al. (1999) where the model building starts at a rather abstract level and ends with a highly detailed model, shown in Figure 10.6.

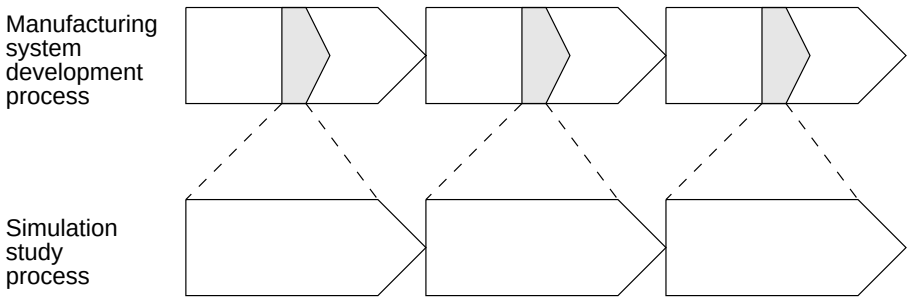


Figure 10.5: *Integrated discrete-event simulation study. Adopted from Holst (2001) and Klingstam (2001).*

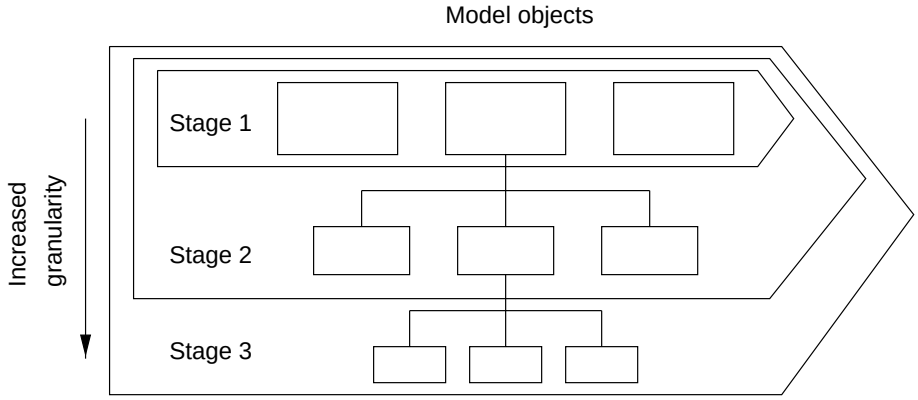


Figure 10.6: *Incremental development of a simulation model with increased granularity.*

In the audit it is evaluated whether stated requirements are complete, consistent and unambiguous. The *functional baseline* is the functional basis of the entire project and thus defines what is to be established. Experience shows that this phase seldom is performed to the level required.

The functional baseline should contain all the information required to proceed with the project. Typical sections in such a document are shown below.

Background The background section contains a short introduction to the project and the reasons for the existence of the project. The purpose is to gain a common view of the reasons for performing the project and put it into context.

Problem statement What is the project to solve? The problem statement should be rather narrow in scope and well defined. Ideally the problem statement focuses on a few measures such as throughput, cost per part or WIP with regard to a set of parameters such as scheduling policies, buffer sizes, product mix etc.

Objectives Objectives are typically to improve the studied system in one way or another, validate its functionality or validate performance before implementation, e.g. increase throughput with 25%, reduce WIP with 25%, validate SFCS functionality, estimate cost per part etc.

Scope The scope declares under what circumstances the objectives will hold. Also considerations about other usages of the simulation model is considered, see Section 10.5.

Organization Documentation of the project organization and the individual tasks.

Project plan Documentation of the expected consumption of resources, nominal time, lead-time and how consumption of resources and time is distributed over time. It should be noted that the expected consumption of resources is only a fair guess at this stage. The project plan may be revised in later phases due to a better understanding of the studied system and its implications on the required effort. When higher levels in the CMM have been reached, better assessments can be made without having to revise the plan in later phases.

Materials and Methods Documentation on how the study will be performed thus including references to process description, methods, and tools to be used, their versions and the environments in which they are to work. This avoids a number of unnecessary problems and must be dealt with especially when there are several developers in distributed and heterogeneous environments.

When performing the functional phase it should be realized that even small changes in the requirements might have large consequences for the model design or implementation. If requirements are unclear and discovered late or if changes of the original requirements are put forward at a late phase in the development process, the total development time might increase considerably. It has been noticed in the performed case studies that, by producing documentation, the simulation study team members are forced to deliver information that is complete, consistent and unambiguous. Verbal information and requirements contain implicit details and are put forward in a context that is lost in later stages of the study. Furthermore, written documents are an efficient way of communicating between team members over time.

It has been experienced that the performance measures commonly used for the studied system are not always appropriate. It should be considered whether the measures really give a good picture of system performance. In some cases measures are used by convention and are not always related to company goals. A classic example that has been experienced in practice, is the factory that is to produce x tons of nails per hour. With a product mix with a large portion of small nails, this measure will be low although profits might be higher for small nails than for large ones. If this is used as a performance measure, profits might suffer considerably after implementation of simulation results. In other words, a simulation study only gives answers to the questions posed in a certain context. If the wrong questions are posed the answers are wrong too.

10.10.2 The Conceptual Phase

The *conceptual phase* is performed to map the system under study into a conceptual model describing the system variables and entities and their relations.

In the methodology proposed here, the documentation is generated before the actual model is built. This effectively avoids the problem of non-existent or

poor documentation. Furthermore, the documentation itself is what the model and the code is generated from, thus facilitating verification and validation.

Vernadat (1996) identifies two different approaches to collect information for enterprise modeling which applies equally well for DES modeling: information collection by group meetings and information collection by interviews. Experience shows that the latter approach provides results of the same quality as the first in a much shorter period of time. In the first case study at BT, see Chapter 8, the first approach was applied due to the training requirements. However, it was discovered that to much time was consumed in that project and in the second case study, see Chapter 9, the number of meetings where all or most parties was gathered, was reduced considerably and so was the development time.

A third approach can be used when DES has been used for some time and Level 3 in the CMM has been reached. When participants in the simulation study are well acquainted with the methods, what information to collect, and how to document that information, the conceptual phase can be performed concurrently by the participants. Such an approach will speed up the process even more.

During the conceptual phase extensive communication with the domain experts is required. The domain experts supply information during the conceptual model phase and verify the conceptual model during the audit. Communication across domain borders is of importance since the domain experts usually have different views of the same entity studied.

As pointed out by Lee (1999*a*), it is useful to establish a set of naming conventions at an early stage. Advantages of using naming conventions are: consistency, ease of identifying entities and ease of collaboration. The naming convention is developed further in the design phase. In the presented case studies, naming conventions was established early in the projects. Similarly it is useful to develop a glossary of terms used when communicating. Sometimes the same terms have different meanings depending on the individual or different terms may have the same meaning. A glossary is a way of avoiding such problems. As DES maturity grows, the glossary will become institutionalized.

When designing new systems it is important that this phase contains a creative process. In the creative process different solutions are proposed and roughly specified. Such a creative process creates a number of possible solutions of the system to be. The solutions are then matched with the functional baseline and results in a few possible scenarios of the system to be. Several scenarios complicate the following steps and phases. However, the scenarios typically has the same components but in different configurations or different components in similar configurations. Modeling several scenarios is thus facilitated considerably by modularization.

In the description of the conceptual model document below, some of the items are defined in more than one place. Where to describe routings are dependent of the tool used and the context. AutoMod, for instance, is part

oriented and the routing is described for the part while QUEST is element oriented and describes the routing for each part at each object. Thus, depending of the kind of simulation tool used, routings are specified either for the parts or for the elements, but not both.

The description of each object in the system does not necessarily have to be fully defined in the conceptual phase. The conceptual baseline should not contain more information than is required for the design phase. The missing information fields are left empty as a checklist for later phases. It is, for instance, unnecessary to specify all times at this stage since such studies requires much resources and could therefore be left out until required. It is, however, of importance that the availability of the information is secured to avoid modifications of the model at a later stage. How the final model is implemented is to a large extent dependent of the input data available. Therefore, this is the right time to find the input data sources, their content, format, etc. without collecting the input data. Especially, an evaluation of the information content is performed to assess e.g. dependencies, which implies that pre-evaluations of data sometimes are necessary.

The system is a complex network of objects that is difficult to describe and document. In the performed projects the conceptual model has been described in textual form. Textual information is comprehensive but difficult to browse especially when projects are large, therefore it is important to have a logical structure of the document, e.g. a structure that follows the material flow. IDEFØ, IDEF3, flowcharts and similar methods can be used to complement the written text. Graphic information is easier to communicate which is important when discussing the conceptual model with e.g. domain experts.

The same set of questions can be posed to several different domain experts, each with their own view of the studied system. It has also been noticed that the same question can get different answers from the same person over time. This can be explained by that posing the question, makes the person think about it a while and thus get a more detailed view of the question. Another reason is probably that, as a project proceeds, new information becomes available to all parties resulting in a new answer the next time the question is posed. This is one of the reasons why it is so important with documentation and that all parties verify documents. The documents and the review of them are a vital part in a *learning organization*.

Temporary Objects

The section on temporary objects (Pidd 1992*a*), usually referred to as parts, should contain a short description of the part and it's purpose, routings, and where applicable, sequences, IAT (Inter Arrival Time), IRT (Inter Request Time), and geometries. When attributes are used they should be described with their type, purpose, and planned usage. Attributes are typically used to control routes, processing times, etc. in the elements processing the part.

Permanent Objects

The permanent objects section should contain a short description of the objects and their purpose.

The object logic is described, possibly including a description of interactions with other objects. The object logic description includes:

- initialization logic,
- process logic, i.e. how to process parts depending on what part, system state, part attributes or object attributes,
- route logic, i.e. where to send parts, and
- request logic, i.e. where to require parts from.

Schedules applied are described as well.

Process descriptions specify e.g. times, restrictions, requirements, inputs and outputs. In the case of transport objects speeds are defined instead of times. The different processes can be cycle, setup, load, and unload.

Failure and repair processes are described by the time scale they are defined on, distribution assumptions or input data sources, and when applicable, labor requirements.

Times and speeds should be described with their dependencies, auto correlation and inhomogeneities if existent.

All attributes used are described with their type, purpose, and planned usage.

System Logic

The system logic describes the permanent objects that in some way affect or control objects, but does not process parts. Each object should have a short description of the object and it's purpose. The detailed description of the object is context dependent and not discussed here. Typical objects are planning systems logic, scheduling systems logic, SFCSs logic, AGV (Automated Guided Vehicle) control systems, material handling equipment control logic, organizational logic, i.e. operator behavior, and interactions with the surrounding system.

10.10.3 The Design Phase

A good strategy during all development is to design a good model structure (as pointed out by e.g. Asklund 1999). That is, limit the dependency between the developers, especially when they are dispersed geographically.

During the *design phase* the overall design of the model is described and the conceptual baseline is filled with model design information, i.e. it is described how conceptual objects will be implemented.

File names and the directory structure is specified and the system is divided into modules, i.e. the implementation infrastructure is defined.

This is a phase that consumes little time, but much rework is avoided in the realization phase, assuming that the design is conceptually verified. Especially, no simulation model building is performed in this phase.

When there are several developers it is obvious that a design baseline is required. A design baseline is also motivated in single developer projects since writing down the design will force the simulation engineer to consider the design and implementation of the model beforehand, which result in a structured model. Furthermore, maintenance of a model is far simpler if the overall structure of the model can be reviewed.

Investigating and designing before coding, not during coding, can avoid a whole lot of work. An experienced simulation specialist will look ahead one step and detect future implementation problems. For instance, being well acquainted with the simulation tool used, details that are difficult or impossible to implement is detected and the design is modified accordingly.

At the lower maturity levels it is not until the end of the design phase it is possible to make a more correct project plan. At Level 4, when the process has been quantified, enough experience has been gathered to make such assessments earlier.

Model Design

The simulation model design baseline typically contains three sections (Babich 1986):

- a decomposition with modules and interfaces,
- descriptions of the modules, and
- descriptions of the interfaces.

Definition of interfaces will allow interchangeability of versions of modules and thereby facilitate CM (Babich 1986), concurrent development and incremental development. Each module is designed in increasing detail until there is a detailed description of each object in the model.

For DES models an interface between two entities in a model or between two modules is typically described with:

Connection The type of connection, i.e. push or pull. Note that the interface description should not contain what objects to connect to. The interface only describes how the connection works.

Parts Definitions of part types that can pass through the interface and the number of parts if batched.

Attributes Temporary object attributes.

Methods If other objects need to determine the state, e.g. current content, means to do that should be supplied. Speaking in object oriented terms this can be referred to as a *method* (Rumbaugh, Blaha, Premerlani, Eddy & Lorensen 1991, Stroustrup 1991).

Variables Common global variables used are described and whether they are read or altered. It should be noted that what is considered good programming avoids global variables to the extent possible.

There are usually interfaces to other systems than simulation models, e.g. data sources. These interfaces also have to be described and documented. An example of an interface to an external data source is given in Chapter 14

Experimental Design

During the design phase not only the design of the model should be performed but also the experimental design with respect to the functional baseline. Experimental design put requirements on the model design and it is therefore suggested to place it in the design phase.

The experimental design in this phase is limited to the details that affect the model design, e.g. what experiments to perform, parameters and variables that should be traced and how they should be traced. The detailed experimental design is done in the experimental phase.

Directory Structure and Naming Convention

It is desirable that a structured and comprehensible naming convention is used. The naming convention is expanded from the incipient naming convention created in the conceptual phase. A well defined structure and a naming convention effectively avoids name clashes and makes all modules, objects and documents easily identifiable especially when distributed and concurrent development is performed.

In the case studies a hierarchical naming convention has been used with a structure as in `ModuleObjectParticular`, e.g. `RbRobot1Cyc` for the cycle process of robot 1 in the robot line module.

The files in a module should be placed in such way that the configurations are stored for all documents in that module. A hierarchical directory structure is easy to use and is attuned with how e.g. CVS stores configurations. The CVS tag mechanism works recursively through subdirectories. Thus, placing all files related to a module in one directory will assure that tagging only affects the files in that module and the sub-modules. An example structure that works well with CVS and the documentation method described in Chapter 11 is shown in Figure 10.7.

```
model/
   documentation
   include_files
   model_files
   source_files
   input_data
   output_data
   macros
   module_1/
      documentation
      include_files
      model_files
      source_files
      input_data
      output_data
      macros
      submodule_1_1
         ...
   module_2
      ...
```

Figure 10.7: *Directory structure example.*

10.10.4 The Realization Phase

During the *realization phase* the model is built, verified and validated. The implementation of the model is just an extension of the design baseline created earlier. With automatic extraction of documentation from the source code documents, see Chapter 11, the design baseline is merely filled with implementation details. In this phase the input data is collected as well.

Building a basic model is easy and usually only takes a few hours, so how come simulation projects tend to consume hundreds of hours? First of all, the modeling time is only a part of the total time consumed in a simulation study (see e.g. Trybula 1994, Umeda & Jones 1997). Secondly, the time to replicate the complex logic commonly found in manufacturing systems tends to take time to model. Thirdly, and most important, debugging tends to consume large amounts of time. It should be noted that the debugging effort is large irrespectively of whether the model is built entirely from the GUI or entirely from scratch with a simulation language. One can argue that the code used when using the GUI is debugged and that the effort therefore should be reduced. That is true to some extent, however, there are several behaviors not anticipated that are well hidden in the built in code.

Verification and validation of the simulation model is made together with domain experts and the customer. It usually only takes a few hours to step

through a model and compare what the customer and domain experts believe is right and how the model is built. If the other phases have been performed correctly there should be no or little changes required.

10.10.5 The Experimental Phase

During the *experimental phase*, execution, analyzes and documentation of results and findings, is performed.

The model is executed to show the customer and domain experts how the model work and early results are presented. The customer and domain experts are now able to come with suggestions on how the experiments should be performed. System behaviors not anticipated might generate new experiment scenarios at this stage. Note that the new scenarios are not supposed to change or add new parameters to be traced defined in the experimental design in the design phase. Not anticipated efforts would be required if additional tracing is required. This is the kind of added work that has been avoided by the careful audits in the previous phases.

Although the basic experimental design was made in the design phase, the details are highly dependent of the dynamics in the simulated system. The number of runs, run lengths and warm-up periods are thus determined at this stage.

Documentation of simulation results should be readable and understandable by *all affected* by the results, including shop-floor personnel when applicable. If the results are presented in an obscure or complicated way, credibility will suffer and implementation might be postponed or never occur.

10.10.6 The Implementation Phase

The *implementation phase* concludes the active work in the project, or subproject if incremental development is employed (Randell et al. 1999). The performance of this phase is outside the scope of this thesis.

10.10.7 The Review and Learning Phase

Whether there is an implementation phase or not, a review of the simulation study should be performed. Issues here can be methods, tools, organization, etc. that are subject for improvement.

When the implementation is concluded an evaluation of real system performance can be compared to the simulation performance estimates. With this information it is possible to enhance prediction validity in future projects.

10.11 Discussion and Conclusions

Most of the time in a simulation study is used for other things than generating a simulation model. Thus the greatest potential of improving the process is

in all the other activities which is at focus here.

The proposed DES methodology is based on the CM methodology and the basic concepts of that methodology has been retained. Applying CM, in this case SCM, has been an invaluable help during the simulation studies. Performing CM using manual routines is possible, but hardly practical.

That quality assurance saves both time and money is a well known fact from manufacturing. Producing a part of bad quality and sending it downstream will cost time due to rework or scrap. The analogy is then that the operations performed in a manufacturing process and the parts produced are in this context the activities in the simulation study and the documents produced respectively. Quality assurance through the continuous audits and reviews is a another basic property of the methodology. The efficient performance of the audits is facilitated through the use of the continuous and incremental creation of documents. The quality of the audits is secured by the participation of the parties in the simulation study.

Another strategy to reduce time consumption has been to move all labor intensive activities, i.e. simulation modeling and data collection, forward in time. Moving those activities forward in time does not only reduce the amount of rework that otherwise might be required, it also moves forward the decision whether simulation will be used for analysis or not. Since it is the same activities that are performed in the early phases whether simulation will be used or not, this decision can be made after the conceptual baseline. However, the early phases are proposed to be performed in a way that makes simulation possible in the later phases, i.e. information on e.g. dynamic behavior should be documented as well. That is, no broad distinction is made between a manufacturing system development process performed with simulation and one that is performed without simulation, see Figure 10.3.

It is not too uncommon that simulation studies are initiated at a late stage. For instance, in the Profilgruppen case study the simulation study started too late diminishing the implementation of some of the results. The conclusion is therefore that it is preferred that the simulation study is initiated when the manufacturing system development process is initiated if the results should be usable. Since both processes share the same activities in the first phases, the conclusion is that there is no reason not to initiate the simulation study from the start. That is, the added effort required to perform a simulation study is not as big as might seem. Most of the activities have to be performed anyway. The added effort is then model building, simulation and analysis.

It is my opinion that a simulation study that is performed and then not used again is to some extent a wasted study. Although great improvements can be made during a simulation study, it is continuous improvements that will result in a state-of-the-art manufacturing system. Simulation can be a powerful tool if such an improvement strategy is employed. Furthermore, since the large bulk of the work have been performed in the first stage of the simulation study, only minor efforts are required to use the model for the manufacturing systems entire life cycle. When performing the succeeding

stages of the study, the same methodological framework is employed.

The presented methodology is incremental in that so-called design objects are turned into configuration items. I other words, each step in the phases generates a requirement for the next step that is then fulfilled and in turn result in a requirement for the next step, and so on until completed. Appendix C presents a trivial example.

The methodology focuses on the phases that precede the realization of the simulation model to reduce the time consumption in those phases as well as reducing the time consumption for realization. However, when studying the time consumed during modeling, experience shows that most of the time is spent on debugging rather than building the model. The methodology does not address this problem since it is highly dependent on what simulation tool used, but the problem is relaxed through peer reviews.

Another strong point in the proposed methodology is the emphasis on reducing communication among team members. This was an expensive experience in the first case study at BT. Documentation and modularization is the key factors affecting the amount of communication required. The communication is moved from the later phases to the early phases. In the early phases all parties are to agree on what is to be done and who is to perform it. In the later phases the actions to be performed are well defined and thus little communication will be required.

As can be seen in Table 10.2, the key process area activities are performed in one or several phases in the DES process.

To be able to work actively with the CMM it is believed that simulation expertise should be kept in a dedicated department. It would be difficult to develop and maintain the expertise if simulation resources were dispersed in the organization. This statement is supported by Hirschberg & Heitmann (1997) that states that the most important factor for successful simulation is professional and competent execution.

It is recommended to start by documenting the *as-is* process instead of the *to-be* process. Mandating top-down a to-be process is a common recipe for failure. If there is a focus on process management, the as-is and to-be processes eventually will converge. The purposes of documenting the process are (Paulk 1998):

- to communicate,
- to understand (if you can not write it down, you do not understand it), and
- encourage consistency.

Table 10.2: *Key process areas mapped into the DES process phases. Not all key process areas are represented as they are in the organization category and are performed over time independent of the projects performed.*

Key process area	*Phases*						
	Functional	*Conceptional*	*Design*	*Realization*	*Experimental*	*Implementation*	*Audit and review*
(2) Repeatable							
Requirements management	•						
Project planning	•						
Status accounting		•	•	•	•		
Subcontract management	•	•	•	•	•	•	
Quality assurance	•	•	•	•	•	•	•
Configuration management	•	•	•	•	•		
(3) Defined							
Integrated management	•						
Simulation engineering	•	•	•	•			•
Intergroup coordination	•	•	•	•	•	•	•
Peer reviews		•	•	•	•		

Chapter 11

Documenting Discrete-Event Simulation Studies

Here a method for documenting the simulation model is presented that is designed as an extension of the methodology presented in Chapter 10. The chapter is based on Randell, Holst & Bolmsjö (2000).

The basic idea for the presented documentation package is that the documentation of a piece of code and the code itself is a unity, i.e. a configuration, and should be in one place to avoid the double maintenance problem. If code and documentation is separated the documentation is bound to lag behind the code development. Furthermore, it is easier to find the documentation for a piece of code if it is in the same place as the code. To facilitate the documentation process a package was developed to extract the documentation from the source files and generate formatted documentation.

The requirements for the described method for documentation of a simulation study are summarized below.

Formats The method should be able to generate hypertext and printable output. During development, hypertext documentation facilitates online retrieval of the latest committed version of the documents. Hypertext documents on the intranet or Internet facilitates development communication in distributed and heterogeneous environments. When a project or baseline is concluded, printed documents are usually required.

HTML can be viewed on almost any computer but generates poorly formatted printed output. PS (PostScript) is good for printed output but can not be used as hypertext. PDF has the benefit of being a hypertext format making it easy to browse and is also formatted to generate a good printed output. Furthermore, PDF files can be viewed on several platforms making it ideal.

Scalable The method should use tools that allow scalable generation of documents, i.e. there should be no upper size limit.

Modular The method should support modular documents and be flexible enough to include or exclude text or documents as the project goes on.

Configurable It should be easy or automatic to generate different versions of the documentation for different usages. Executives need a short summary of the project while developers need an extensive documentation with all the details both for quality assurance and for later maintenance.

Simple The tools used for documentation should preferably be well known to the developers and if not, at least easy to learn. The generation of the different formats should be easy and fast so that the latest versions always can be kept updated.

Heterogeneous In the industry today most of the computers used are PCs running MS-Windows of some flavor. However, there are other operating systems and platforms used making heterogeneous environments an issue. The portability issue can be divided into generation of documentation and viewing documentation. If possible, the method for generating the documentation should be portable. The generated output formats should be portable between different environments for easy distribution of documentation.

One source The documentation should be generated from one source, i.e. no *cut-and-paste* should be required, thus avoiding the double maintenance problem.

Transparent The tools used to generate the documentation should be transparent to the development environment used.

11.1 Documentation Method Properties

LaTeX and add on packages were used to tackle the documentation requirements. To generate the LaTeX inputs the CodeDoc tool was developed. Using different packages allowed automatic generation of HTML, PS and PDF documents. The Make (Feldman 1979) utility was used to keep track of the configuration and for building documentation.

All the tools and packages used together with CodeDoc are shareware distributed under the GNU Public License. As other GNU software they are of high quality and are available on almost any imaginable platform.

In the developed method each developer responsible for a module is assigned a module directory where the Makefile is placed, see Figure 10.7. The Makefiles are arranged hierarchically and any portion of the documentation can be generated from any level by working down the hierarchy. The Make utility only generates new documents for the sources that are updated since the last document generation, reducing build time.

The approach is modular where the modules and interfaces are described by the directory name, the document name and conditional compilation flags.

The top level does not need to know how the sub levels have implemented the generation of the documentation. The top level only need to know what files to include and how to generate those files. The same holds for each lower level in the documentation hierarchy.

The generation of the documentation is scalable and as many files of documentation and sources as needed can be added to form the final documentation.

Conditional compilation can be used to generate the entire project documentation in detail or a short summary.

11.2 The CodeDoc Tool

CodeDoc was developed to allow the code and the corresponding documentation resides in the same file, i.e. the source code. The basic ideas are similar to DocStrip that is used to document and generate LaTeX packages (Mittelbach, Duchier, Braams, Woliński & Wooding 1999).

The developed CodeDoc tool is transparent to the development environment. The transparency is managed by using the comment characters of the simulation or programming language used, in the case of SCL, '`--`', and in the case of C/C++, '`//`'. The comment characters of the language used are set using a command line option.

The syntax of the CodeDoc tags is (with SCL comment characters):

`--.`	Ordinary LaTeX source
`--(entity)`	The `--( ... )` sequence defines e.g. a procedure, variable, constant or attribute
`--/HEADING/`	The `--/ ... /` sequence defines a sub heading.
`--{` `--}`	The `entity` code is enclosed by two curly braces. This code is excluded when the `-v` (verbose) command line option is not given.
`--[` `--]`	Two brackets enclose code that is not excluded when using the `-v` command line option is not given.
`--<condition_flags>`	The current line is executed only if the flags are set
`--<condition_flags`	Conditional execution begins with this line
`-->condition_flags`	Conditional execution ends with this line

The `condition_flags` tags are currently not implemented.

An example of how a piece of documented SCL code would look like is shown in Figure 11.1. The resulting formatted output is shown in Figure 11.2. Note that each entry is automatically included in the table of contents and the index.

```
--(ExampleLogic)
--.\verb|ExampleLogic| performs a ... and returns ...
--.
--/SYNTAX:/
--[
--.routine ExampleLogic(InVar : type) : integer
--]
--.
--/ARGUMENTS:/
--.\verb|InVar| is a ...
--.
--/EXAMPLE:/
--[
--.var
--.  RetVal : integer
--.begin
--.  RetVal = ExampleLogic(InVar)
--.end
--]
--.
--/USES:/
--.\crossref{SomeRoutine}
--.
--/CAVEATS:/
--.This procedure only works when ...
--.
--/AUTHOR:/ Lars Randell (lrandell@robotics.lu.se)
--.
--/CVS:/ Revision: 1.5
--{
routine ExampleLogic(InVar : type) : integer
var
  LocalVar : integer
  OutVar : integer
begin
  -- inline comment
  LocalVar = SomeRoutine(InVar)
  ...
  return(OutVar)
end
--}
```

Figure 11.1: *Example of documented source code.*

`ExampleLogic`

`ExampleLogic` performs a ... and returns ...
SYNTAX:

```
routine ExampleLogic(InVar : type) : integer
```

ARGUMENTS: `InVar` is a ...
EXAMPLE:

```
var
  RetVal : integer
begin
  RetVal = ExampleLogic(InVar)
end
```

USES: `SomeRoutine`
CAVEATS: This procedure only works when ...
AUTHOR: Lars Randell (lrandell@robotics.lu.se)
CVS: Revision: 1.5

```
routine ExampleLogic(InVar : type) : integer
var
  LocalVar : integer
  OutVar : integer
begin
  -- inline comment
  LocalVar = SomeRoutine(InVar)
  ...
  return(OutVar)
end
```

Figure 11.2: *Example of formatted CodeDoc output generated in verbose mode, i.e. with the code included.*

11.3 Discussion and Conclusions

The documentation extraction tool is developed for plain code. Many developers consider it simpler to develop models using a GUI. My view is quite the opposite, at least when the GUI is a tool for building models instead of a modeling tool. Well-structured code is easier to browse and comprehend than trying to comprehend a model by using the GUI. Similarly it is easier to build and comprehend a model using a programming language such as BCL. The entire model is then easily viewed by browsing the code. The same documentation tool can then be used to document the BCL code and the SCL code and thus can the entire simulation model be documented.

The right place for documentation of a simulation model is as 'close' as possible to the object the documentation tries to explain. Furthermore, this documentation should be simple to include in a simulation study report. The support for such a strategy is, however, quite weak in simulation tools.

Whatever tool is used to facilitate the documentation of a model, in the end it is the developers that has to change the documentation as they change the code. Documentation of what has been done is not an effortless task and requires discipline.

To fully integrate and utilize the documentation method the documents generated in the conceptual, design and realization phases are partly stored as documented source files that are continuously filled with more and more information and finally the code.

The developed tool for documentation of sources has been used in a few cases for documentation of SCL and C/C++ code. It has been experienced that it is feasible and desirable to integrate code with the corresponding documentation. However, the tools are not simple enough for mainstream use, thus the next step in the development of the methodology would be to build a GUI that keeps track of the documents, their configuration and generates the documentation with a few clicks. Another problem is that of using LaTeX for generating the documentation. LaTeX is not a common tool for writing reports. However, the principle the tool is based on, i.e. using tags in comments, is generalizable. It should be possible to build a similar filter to extract documentation for e.g. MS Word using a Visual Basic script.

Since the code and the documentation of it is in one unit it is simpler to reuse code and documentation in other projects. When the code is reused in another project, the documentation of it is in place as well.

Chapter 12

Data Collection Framework

In Chapter 7 it became evident that most of the problems with producing a high accuracy simulation model could be referred to the poor accuracy of the input data.

In this chapter a classification of losses will be made that is used to define time-scales. The different time-scales are used in different context to define the failure processes and availability measures. The framework is then used to specify how input data should be collected and analyzed.

12.1 Data Usage

This chapter should be viewed in the context of the life cycle of the manufacturing system, shown in Figure 12.1. Collection of data from the manufacturing system can, and should, be used actively in many of the stages of the life cycle. The goal of the data collection framework is to collect data that can be used for a number of activities, of which one is input data for DES. Other usages are online monitoring of the manufacturing system performance, retrospective manufacturing system performance analysis, and continuous improvements of the manufacturing system. Data for different usages require data to be collected with high granularity.

Managers need information on the system level to be able to monitor system performance. They might also need information to set wages according to some wage system.

Production engineers and operators require statistics to continuously improve the manufacturing system and detect upcoming problems. Interviews with production engineers have revealed that they would like to study data at the sensor level to detect problems and their causes. Here it is proposed that the causes of losses be collected which would provide the same information.

Manufacturing system developers should be able to predict cell and manufacturing system performance of manufacturing systems using data from

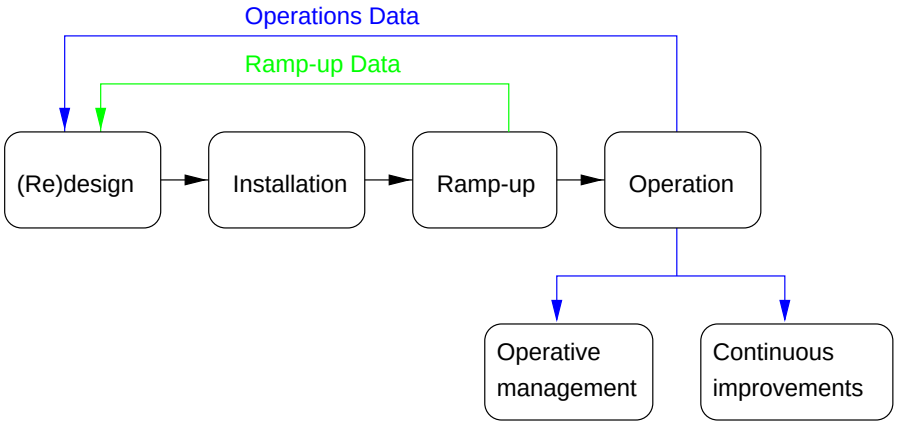


Figure 12.1: *Usages of data in the manufacturing system life cycle.*

existent systems. Data is used both for design of entirely new manufacturing systems as well as for redesign of existent systems.

It became evident when analyzing the manufacturing system at Volvo that it is important to take into account the dynamic behavior. The manufacturing system was well balanced with little slack when using a static analysis. However, when the different failure processes were associated with the cells the system became unbalanced due to different availabilities, hence the requirement of reliable statistics from similar manufacturing systems.

In discussions with Bengt-Göran Olsson at Volvo Car Corporation in Gothenburg, it was concluded that data has to be collected automatically to avoid the unclassified stops. Ironically, Ingemansson (2002) concludes that the automatic data collection should be complemented with manual entries. Short stops are quicker to solve than register. However short the stoppages are, they have a large effect on performance and should thus be collected (De Smet et al. 1997, Kuivanen 1996). In conclusion, a data collection system should be automated to the extent possible and manual entries are used when necessary. The manual entries could be facilitated in that the system could propose a number of possible stop codes based on the sensor information available.

Before input data becomes usable in a DES model the following steps has to be performed;

Collection Input data is collected in the system under study at the granularity level required.

Filtering Input data is then filtered or extracted to generate data samples that can be used for input data analysis.

Analysis Parameter estimations and goodness of fit tests are performed to generate appropriate distributions to be used in the simulation models.

If this step does not succeed, empirical distributions or file driven distributions can be used.

12.2 Dependencies

The most common problem with input data is that it contains dependencies. One of the major points of the presented framework is to generate input data that does not have as many dependencies, i.e. it is more general than data normally collected. Consider the data collected in the case study described in Chapter 7. The TTR times included the time waiting for the operator to arrive, i.e. there was a dependency between the repair times and the number of operators. Furthermore, and perhaps more serious and deceiving, is the fact that there will then be dependencies with all repair processes in the system where an operator is required. If, for some reason, one of the repair time distributions is changed the workload on the operators will change, and thus will the wait time change for all repair processes in the system where operators are required. To conclude, data collected with these kinds of dependencies will only be useful when exactly the same system is modeled as where data was collected from, i.e. data will not be as useful if the system is modified in any way, which usually is the purpose of a simulation study.

12.3 Losses

When collecting data the causes of the losses are to be determined to the extent possible. Examples of states are:

- busy
- idle
  - idle due to starvation
  - idle due to being blocked
  - idle waiting for a signal from another entity or a control system
- setup
- down
  - wait for repair
  - repair
- tool change
- manual stop
- safety stop

- preventive maintenance
- off shift or break
- shut off

The idle and busy states are required for correct analysis of the failure distributions on the operative time scale. This is also the reason for including the safety stop state. The wait for repair state is required to separate the wait time from the actual repair time, which is used when the impact of the number of operators and safety zone sizes are to be analyzed, see Section 12.5.

Losses can be classified along three dimensions. Firstly, losses can be in control or out of control. The level of control depends on the cause and the owner of the cause. Secondly, losses can be planned or unplanned. Thirdly, losses are dependent of either a time scale or the number of cycles. The latter is extremely important to be aware of, or the failure processes will be modeled incorrectly and is thus the motivation for Section 12.4.

Typical causes of *planned losses* are: break, meeting, setup, and preventive maintenance.

Synchronization losses can have the states: blocked, starved, and wait signal. Synchronization losses can be classified as static or dynamic. *Static synchronization losses* are a result of static differences in average cycle times between succeeding operations, usually referred to as *slack*. *Dynamic synchronization losses* are generated by e.g. cycle time variance or delays generated by stops in upstream or downstream operations. Since dynamic synchronization losses are dependent of several dynamic effects in the system, both upstream and downstream, they can not be calculated analytically.

Unplanned losses independent of the busy time e.g. material starvation on the line level, are handled separately since they usually are out of control and not used for analyzes.

Unplanned losses dependent of the busy time can be said to be under control in the long-term perspective. Typically preventive maintenance activities will reduce losses and so might other productivity improving activities such as reduction of setup times (Ericsson 1997). Failures are unplanned and usually based on the busy time of an entity.

12.4 Time-Scales

In this section we will start with *wall-clock time*, subtract losses and end with the busy time used for production. Each subtraction of a class of losses describes a new time-scale, shown in Figure 12.2. The different time-scales are used for different purposes and can be redefined at will. To be able to use the different time-scales in practice the states presented in Section 12.3 has to be collected.

Ingemansson (2001) and Nord, Pettersson & Johansson (1998) present similar classifications of losses. However, what is presented here is a framework

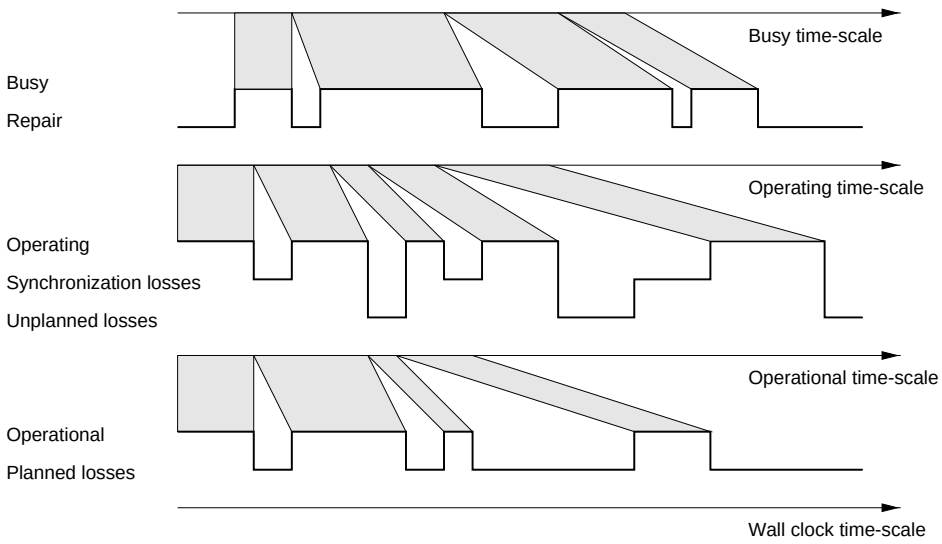


Figure 12.2: *Time-scales.*

for how to collect data from a manufacturing system and extract times from a database for usage with e.g. DES. By collecting detailed operations data from a manufacturing system, time vectors for different classes of losses can be generated for different purposes depending on how the system should be modeled.

In order to put the time-scales in perspective in the context of the commonly used availability measures, Appendix B presents different availability measures. They are presented because the way they are extracted is different from the way data is extracted to generate the time-scales. Availability measures are used to assess a component's maintainability. However, when assessing the total systems performance the losses included in the availability has to be complemented with other losses as well. In summary, availability measures are used to compare different competing components, while e.g. throughput is used to assess the total system performance compared to another competing systems performance. Both rely on the same set of data that is treated the same way, but used differently.

12.4.1 Notations Used

A set of states is written as

$$S_j = \{s_1, s_2, \ldots, s_n\} \subset S_\Omega$$

where S_Ω is the entire set of states defined for the system.

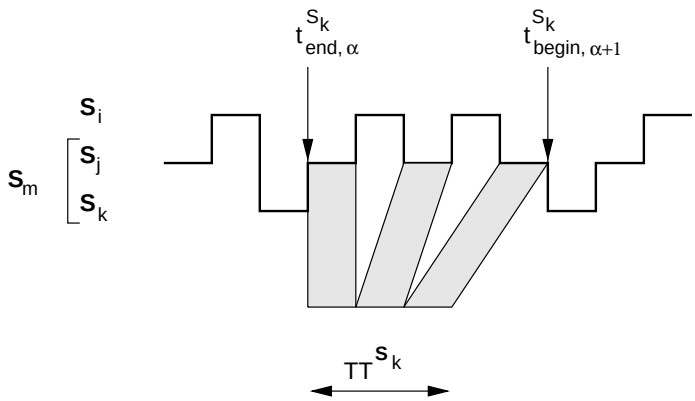


Figure 12.3: *Time Between for a set S_k on a time-scale defined by the set S_m.*

Specific points in time are denoted

$$t^{S_j}_{k,begin/end}$$

where the superscript is a set of states, $S_j \neq \emptyset$, that might be a single state. The subscript k is an index and *begin/end* is a specification whether the state is entered or exited.

Time pairs that specifies when a set of states, S_j, have been entered and exited is written as

$$t^{S_j}_k = \left(t^{S_j}_{k,begin}, t^{S_j}_{k,end}\right)$$

An ordered vector of time pairs in a set of states, S_j, is written as

$$\mathbf{t}^{S_j} = \left[t^{S_j}_1, t^{S_j}_2, \ldots t^{S_j}_n\right]$$

The period of time in a set of states, S_j, is written as

$$T^{S_j}_k = t^{S_j}_{k,end} - t^{S_j}_{k,begin}$$

and the vector of time periods is written as

$$\mathbf{T}^{S_j} = \left[T^{S_j}_1, T^{S_j}_2, \ldots T^{S_j}_n\right]$$

On a time-scale described by a set of states, S_m, time between two successive set of states in the set $S_k \subset S_m$ can be written as

$$\mathrm{TB}^{S_k} = \sum T^{S_m}_i; \quad t^{S_k}_{\alpha,end} \leq t^{S_m}_i \leq t^{S_k}_{\alpha+1,begin} \tag{12.1}$$

An example is shown in Figure 12.3

The filters written in Perl, see Section 7.4, was based on Equation 12.1 to extract data samples on different time-scales.

12.4.2 Wall-Clock Time

The wall-clock time set, S_Ω, is the junction of all the time subsets arranged in a hierarchical structure. Wall-clock time can be divided into time scheduled for production and unscheduled time. These are more of administrative interest and will not be discussed further. Instead we focus on the operational time, i.e. the time left when planned losses have been subtracted.

$$\mathbf{T}^{S_\Omega} = \mathbf{T}^{S_{operational\ time}} + \mathbf{T}^{S_{planned\ losses}}$$

12.4.3 Operational Time

The *operational time* is the time available for production and is defined as

$$\begin{aligned}\mathbf{T}^{S_{operational\ time}} = & \mathbf{T}^{S_{synchronization\ losses}} + \\ & \mathbf{T}^{S_{unplanned\ losses\ independent\ of\ busy\ time}} + \\ & \mathbf{T}^{S_{operating\ time}}\end{aligned}$$

The operational time-scale is used to extract TBF dependent on the time a system is switched on or operational, but not necessarily busy. However, there might be implementation problems depending on whether this time scale is defined in the simulator or not. This time-scale is also suitable for productivity measures since it is the time that the system is supposed to be busy.

12.4.4 Operating Time

The operating time scale is used to extract TBF dependent of the busy time. On this time-scale an entity alters between the repair state and the busy state. The operating time is defined as:

$$\begin{aligned}\mathbf{T}^{S_{operating\ time}} = & \mathbf{T}^{S_{unplanned\ losses\ dependent\ of\ busy\ time}} + \\ & \mathbf{T}^{S_{busy\ time}}\end{aligned}$$

12.4.5 Busy Time

Most stops are dependent of the busy time.

$$\mathbf{T}^{S_{busy\ time}} = \mathbf{T}^{S_{logistic\ losses}} + \mathbf{T}^{S_{process\ time}}$$

The remaining measurable losses are then logistic losses, e.g. movements. Although necessary things such as movements can not be removed entirely, they can be reduced.

There is a point in measuring both these times since it indicates if it is the process that has changed or if it is the devices surrounding the process when

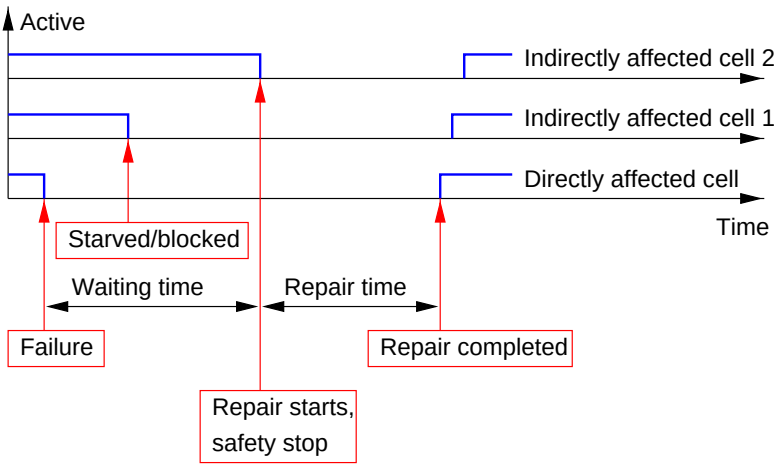


Figure 12.4: *Indirect losses in a safety zone.*

an increased cycle time is detected. Adding an online trend tracking system to these times will facilitate the detection of upcoming problems.

12.4.6 Process Time

The *process time* is the time an entity is processing, i.e. adding value, e.g. the cutting time in a lathe or the weld time in a welding cell.

The process time itself might also be improved, e.g. through high speed welding or high speed machining.

12.5 Safety Zones

When a safety zone consists of several cells the other cells within the zone are affected indirectly when the safety zone is entered to perform a repair, shown in Figure 12.4. Cells upstream and downstream are affected either because they are starved or blocked or because a safety stop is caused by an operator entering the zone for repair.

The number of cells in a safety zone as well as the number of operators become a design parameter and it is thus desirable to model the correct behavior in the simulation model. Starved and blocked states are handled automatically in a simulation, but modeling failure processes for a cell that after a stochastic period of time stops the cells within the safety zone is not supported by simulators.

To attend to the problem of coupled stops the failure and repair logic of QUEST was rewritten. The default behavior of the repair logic is to require labor if required, and then put the element in the repair state for the specified

Table 12.1: *Data collection fields with examples for one entity.*

Date	*Time*	*Entity*	*State*	*Cause*	*# Processed*	*# Scrapped*	*# Reworked*	*Begin*	*End*
2002-04-10	08:56:27	4802	Off	9824	0	0	0	0	0
2002-04-10	08:58:30	4802	Busy	0	0	0	0	1	0
2002-04-10	08:59:30	4802	Busy	0	1	0	0	0	1
2002-04-10	08:59:30	4802	Idle	Blocked	0	0	0	0	0
2002-04-10	09:01:27	4802	Idle	Starved	0	0	0	0	0
2002-04-10	09:01:58	4802	Busy	0	0	0	0	1	0
2002-04-10	09:02:48	4802	WT	5607	0	0	0	0	0
2002-04-10	09:06:23	4802	Repair	5607	0	1	0	0	0
2002-04-10	08:59:48	4802	Busy	0	0	0	0	1	0

duration of time. The logic was rewritten to require labor and when the labor arrived to perform the repair, the other elements in the safety zone were halted in the repair state. When the repair was completed all elements within the safety zone was put in ready state again. To do this each element belonging to a safety zone had an attribute that designated what safety zone it belonged to. The repair logic searched the element list to find those elements and then scheduled failure interrupts for those elements. The pseudocode is shown in Figure 12.5.

Note that personnel that performs repairs has to remain by the cell until the system is up and running to verify the repair. Usually when modeling repairs with personnel they are released immediately after the completed repair and the real behavior is difficult to model.

12.6 Database

In summary the required fields in the proposed data collection framework would include date, time, entity id, state, number of processed parts, scrapped parts, reworked parts, part identity, and cause. To detect speed losses each start and end of a work cycle should be stored. An example is shown in Table 12.1

12.7 Input Data Generalization

As in the case of Volvo, many of the manufacturing systems might have similar configurations making possible generalizations of interest.

As presented above the smallest entity for which data is collected is a cell.

```
procedure zone_repair()
begin
  -- get the repair process
  repair_proc = get_curr_repair_proc()
  -- require labor if required
  pid = do_process_requirements(repair_proc)
  -- get the user interrupt
  the_uintr = get_user_interrupt(celem->zone_name)
  -- get repair time
  rep_time = sample_cycle_time(repair_proc)
  -- create a BCL command to set the repair time
  BCL_str = "SET USER INTERRUPT '" + \
    celem->zone_name + \
    "' ATTRIB 'rep_time' TO " + \
    str('%g', rep_time)
  -- execute the BCL command
  bcl_ret = BCL(BCL_str)
  -- create a safety zone element list
  zlist = list_create()
  -- make a list of the elements in the safety zone
  make_zone_list(celem->zone_name, zlist)
  -- for each element in the safety zone, schedule a
  -- repair interrupt
  schedule_interrupt(the_uintr , zlist, 0)
  -- do the repair in current element
  do_repair rep_time
end
```

Figure 12.5: *`zone_repair` is called by `zone_failure` to start the repair of an element. Elements within a safety zone, i.e. those elements that will stop as labor enters the safety zone, are associated with the safety zone through an element attribute. The `zone_failure` logic is rather large and only minor modifications were made and it is therefore not presented here.*

However, by collecting data on a lower level, i.e. the devices comprising the cell, data can be made more generalizable. Each cell's configuration might be different in another manufacturing system. By collecting failure data on the device level, the new configuration's failure process can be assessed using the aggregated failure processes of the devices of which it consists, provided the stochastic variables are independent. Here that is performed by associating each cause with a device, and thus can the failure characteristics of each device be assessed.

When assessing the failure processes for the devices the time-scales have to be taken into consideration again, but now on yet another level. The busy time for a device might differ from the cycle time in which it is a component. Thus is there a time scale for the device. This might be to take things a little bit too far to be practical, but one should be aware of the problem.

In the case study presented in Chapter 7 it was hard to find any patterns in input data. There are a few dependencies that might affect TBF and TTR characteristics when trying to generalize.

The repair time is dependent of the skill level (as discussed by e.g. Ilar 1997). The TBF is dependent of how mature the process is, i.e. new processes are more prone to production stops than old and well established processes.

Another issue is that the focus of improvement activities is most likely focused on the worst performing cell, which results in a not fully exploited performance potential in the other cells. That is, the system is improved with the total system performance in mind. When developing another manufacturing system the not fully exploited cell configuration might become the bottleneck. The new system will then be assessed using data from a system where the cell performance was not fully exploited.

The stop characteristic of a device is dependent of the individual device, the process and the configuration of the cell. That is, when the same type of device is to perform the same operation on another part, it might not have the same failure characteristic.

12.8 Discussion and Conclusions

There is a rule in simulation saying 'Shit In Shit Out' (SISO), i.e. if the input data is corrupt, so are the results. Usually it is difficult to check the validity of the simulation model and compare it to a real system with accuracy. In the case study performed at VCBC such a comparison was possible and it became painfully evident that input data quality is vital for high accuracy simulations. One of the main reasons, besides time consumption, for not using DES at ABB BiW and VCBC, was the lack of accurate input data. It was for the kind of highly accurate simulations required at ABB BiW and VCBC that the proposed data collection framework was developed.

The main advantage of the detailed input data collection is that it can be used for several analyzes, not only for DES. If such a complex system were to

be implemented for the purpose of DES only, there would be little motivation to do so. However, with the multiple usages, the added cost of implementing the more detailed data collection is motivated.

The 'six big losses' presented by Nakajima (1988) can all be handled with the presented framework. That is, the down time, speed losses and defect products classes of losses are all taken care of.

Another advantage of the detailed data collection is that dependencies are removed. Such dependencies complicate model building and make results less reliable.

Chapter 13

Parameterized Simulation Models

The case study described in Chapter 7 was later developed further as a master's thesis work (Pålsson & Quist 2002) supervised by the author and Lars Holst.

In Chapter 7 it was established that making a simple model would not mimic the systems behavior in an appropriate way and the complex model required too much resources. The complex model was not modularized and reusing the simulation model was therefore difficult. Furthermore, maintaining the code would prove difficult. A method for modularized and quick development of DES models for the studied manufacturing system type is therefore presented here.

13.1 Components

By modularizing the simulation model code it was possible to parameterize the simulation models, i.e. most parts of the simulation model would be possible to describe with a set of parameters. Many of the production lines at the plant at Volvo were similar in their structure and logic, which enabled the modularized code.

13.1.1 Spreadsheet

To build the model an ordinary spreadsheet generating the BCL code to build the model was developed. The spreadsheet also generated the SCL code used to initialize the model. The spreadsheet was composed of a number of sheets, each dealing with a set of model constructs:

Elements The elements sheet contained the element names, element classes, e.g. machine or buffer, geometry, x, y, and z position, element process logic name, element route logic name, element request logic name, element initialization logic name, and the number of cycle processes.

Failures The failure sheet defined the failure and repair process for each element by specifying the time scale, i.e. simulation time or busy time mode, the failure distribution and the repair distribution with parameters.

Processes The process sheet defined the cycle times for each process associated with an element.

Connections Element connections, i.e. possible routings, were defined in the connections sheet.

Parts The parts sheet defined the parts and their geometries.

Sources There was also a sheet that defined what sources to compile.

Signals Finally there was a sheet defining all the signals and attributes of the elements that was used by the developed generalized logic.

The generated BCL scripts was then copied into the model generation script that generated and compiled the model. The SCL code was copied to the simulation initialization logic and the constants include file.

13.1.2 General Logic

The fundamental base for the generalized logic was variable passing by the use of element attributes. This diminished the need to define element specific logic, constants and variables that were required in the case described in Chapter 7.

The basic logic was the client server logic that then was used to build other logics. A client could only have one server while a server could have several clients. When there was a merging material flow the server logic was put on the input side of the merging element. When there was a splitting material flow the server logic was put on the output side of the splitting element. The same logic could thus be used for all kinds of material flows where handshaking was required.

The client server logic, shown in Figures 13.1 and 13.2, works as follows:

- The client requires service from the server by sending a request signal;
- The server selects the first request in the request queue and acknowledges the request by sending an acknowledge signal when ready;
- The client locks the server to be it's own and then performs a cycle. The locking is performed by sending a lock signal;
- The client unlocks the server by sending an unlock signal allowing other clients to access the server.

```
procedure server_handshake(client_elem : element)
begin
   -- wait until unlocked before executing requests
   wait_unlocked(client_elem)
   :WaitNewRequest:
   -- wait for request signal from client
   wait until signal client_elem->req_sig
   -- If there are no clients waiting then set signal
   -- to OFF.
   if(NOT sig_wait_count(client_elem->ack_sig)) then
      signal client_elem->req_sig OFF
      goto WaitNewRequest
   else
      -- get first waiting client
      sig_entry = get_sig_wait_entry(client_elem->ack_sig, 1)
      -- signal acknowledgment to client
      signal client_elem->ack_sig for sig_entry
   endif
end
```

Figure 13.1: *Pseudo code for the* `server_handshake` *routine.*

```
procedure client_handshake(server_elem : element)
begin
   -- signal request to server
   signal server_elem->req_sig ON
   -- wait for acknowledgment
   wait until signal server_elem->ack_sig
   -- lock server from others
   lock_elem(server_elem)
end
```

Figure 13.2: *Pseudo code for the* `client_handshake` *routine.*

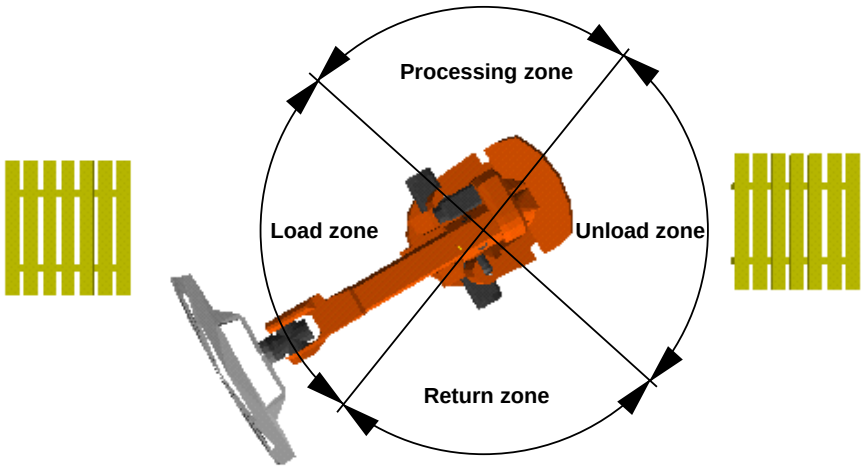


Figure 13.3: *The robot zones where handshaking was required.*

Most robot cells had several zones, e.g. load zone, processing zone, unload zone, and return zone, see Figure 13.3. Before entering a zone the access to that zone had to be acknowledged to avoid several robots entering the same zone at the same time. The client server routines made this possible. A standard process logic, shown in Figure 13.4, worked as follows:

- request access to load zone,
- load the part and acknowledge that the zone have been exited,
- process the part,
- request access to the unload zone,
- unload the part and acknowledge that the zone have been exited, and
- return to the original position.

13.2 Synchronized Transport Logic

In one part of the system there was a synchronized transportation mechanism between manufacturing cells, shown in Figure 13.5. This logic is slightly more complicated since a controller is to synchronize the transport cycle between a number of elements. Each position in the transportation line in turn has to

```
procedure mach_split_merge_prc()
begin
   -- require acknowledgment from input element
   server_handshake(input_elem)
   -- do the load cycle
   execute_cycle(LOAD_CYCLE)
   require part ANY
   unlock_elem(input_elem)
   -- do the process cycle
   execute_cycle(PROCESS_CYCLE)
   -- require acknowledgment from output element
   client_handshake(output_elem)
   -- do the unload cycle
   execute_cycle(UNLOAD_CYCLE)
   -- transfer the part
   pass()
   unlock_elem(output_elem)
   -- do the return cycle
   execute_cycle(RETURN_CYCLE)
end
```

Figure 13.4: *Pseudo code for a typical machine that, in this case, has server input and client output.*

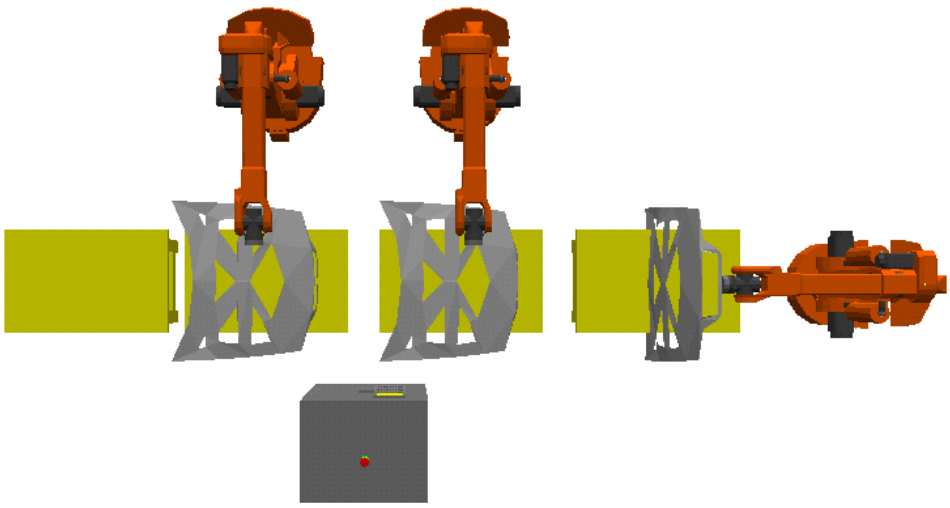


Figure 13.5: *The transfer stations, controller and robots in a synchronized line.*

```
procedure ctrl_handshake(the_elem : element)
begin
  -- tell the controller that it is ready to transfer
  signal celem->rdy_transf ON
  -- wait for transfer signal
  wait until signal the_elem->start_transf ON
  -- do the transfer
  execute_cycle(G_MOVE_FORWARD_CYCLE)
  -- signal that the transfer is ready
  signal celem->transf_rdy ON
end
```

Figure 13.6: *Pseudo code for the* `ctrl_handshake` *routine used in the transfer stations process logic shown in Figure 13.8.*

communicate with external elements, in this case robots processing, loading or unloading parts.

The same basic signaling mechanisms was used here as well, and all parameters required was assigned as element attributes to make possible a generalized logic.

The functionality, shown in Figures 13.6, 13.7, and 13.8, is as follows:

- Each transport element sends signals to the controller that they are ready to transport;
- When the controller have received signals from all the transport elements it returns a transport signal to perform the synchronized transport;
- The transport elements performs the transport cycle and then sends a signal to the elements processing, loading or unloading parts;
- The process, load, and unloading elements performs a cycle and then sends a signal to the transport element to acknowledge that the processing is finished;
- When the transport element receives the process ready signal, the cycle can start all over again.

13.3 Modeling Method

To model a new system the method described in Chapter 10 is used. In the realization phase the following steps are conducted:

1. Define the model constructs in the spreadsheet.
2. When possible, use the readily available logic for the elements. If customized logic are required, use the readily available logic as a template.

```
procedure buf_handshake(the_elem : element)
begin
  -- wait for ready to transfer signals
  for ii = celem->first_rdy_transf to \
           celem->last_rdy_transf do
    wait until signal ii ON
    signal ii OFF
  endfor
  -- signal transfer
  signal celem->start_transf ON
  -- wait until transfer is ready
  for ii = celem->first_transf_rdy to \
           celem->last_transf_rdy do
    wait until signal ii ON
    signal ii OFF
  endfor
end
```

Figure 13.7: *Pseudo code for the* `buf_handshake` *routine used in the transfer line controller.*

```
procedure transfer_elem_prc()
begin
  require part ANY
  -- wait until robot is ready to perform the process
  server_handshake(celem)
  wait_unlocked(celem)
  -- tell the controller we are ready to transfer
  ctrl_handshake(ctrl_elem)
  pass()
end
```

Figure 13.8: *Pseudo code for the* `transfer_elem_prc` *routine used in the transfer stations. Note that the transfer stations communicates with both the controller and the processing element.*

3. Cut and paste the BCL code into a macro file.
4. Cut and paste the SCL code into the simulation initialization logic.
5. Build and compile the model by executing the BCL macro.

13.4 Discussion and Conclusions

With the developed tools it is fairly simple to model a new manufacturing system of the studied type. Furthermore, the development time is reduced considerable without having to reduce model accuracy. The modeling time is believed to have been reduced to 20–50 hours. The key to this is the use of general logic with parameters passed as attributes.

The developed method is limited in that only models of manufacturing systems of the studied type can be developed. However, the basic concepts of parameterization and modularized and generalized code can be utilized in similar contexts.

Chapter 14

Integrated Factory Simulation

Here an integrated approach to build large simulation models is presented. The simulation model is built from, and driven by, a database reducing the model realization and input data collection efforts. The chapter is based on Randell & Bolmsjö (2001).

In general the most time taking phases of a discrete-event simulation study is the input data collection and model building (Banks et al. 2000, Bernard 2000, Umeda & Jones 1997, Liyanage & Perera 1998, Trybula 1994, Heitmann et al. 1997, Hirschberg & Heitmann 1997). Building large DES models of entire factories with traditional methods is thus time consuming and the required maintenance effort is bound to lag behind. A better solution would be to have model information stored in a continuously updated database. However, creating and maintaining such databases is also time consuming. To motivate such databases the information should be shared by several applications, thus reducing the total information management time for the applications sharing the information. The method proposed here reuse readily available manufacturing system information without creating new databases; instead information stored in an ERP (Enterprise Resource Planning) database is reused.

The basic problem with large simulation models is the complexity of the model logic and the amount of input data required. Conventional methods embed data and logic into the simulation model, which works well until the size of the simulation model reaches a certain size. This size is reached when the simulation model cost reaches the potential cost savings, the system under study changes before the model is ready, the results arrive after the point of usage (Ball & Love 1994) or when the maintenance effort exceeds practical limits. When this size is reached other methods have to be considered.

14.1 Planned Usage

The set of tools developed was planned to be used for testing master plans a few months ahead for the BT plant in Mjölby. The plant at Mjölby has 1'000 employees and produces approximately 35'000 warehouse trucks a year. To test master plans the entire factory had to be modeled and all products produced in the factory had to be simulated.

The simulation was to be fed with a master plan that then was broken down to production orders. The simulation was to be executed to find bottlenecks and delayed deliveries. By identifying the bottlenecks, possible countermeasures could be implemented in the simulation database before the simulation was executed again. Since the simulation was entirely driven by a database all modifications to the simulation model was performed in the database that then was used to generate the modified simulation model. Possible countermeasures were e.g. added shifts, outsourcing to suppliers or added resources. The impact of future investments in new resources could also be tested. It was estimated that approximately four experiments would be performed during a day before establishing a final master plan. It was considered that not more than one day could be spent on this activity each month.

14.2 Requirements

The following requirements were considered;

Time The method should reduce development time and project lead-time.

Modeling Creating and running a simulation model should require little or no manual intervention.

Software independent The database should be built and maintained independently of the DES software used. The DBMS (DataBase Management System) should also be exchangeable.

Reusable The method should be transferable from one project to another.

Modular The method should be modular to support the reusability requirement.

Scalable The method should be applicable for large or even very large enterprises.

Heterogeneous The method should work in heterogeneous environments. Furthermore, the software should be portable in between different platforms.

Geographically transparent The method should be transparent to geographical separation.

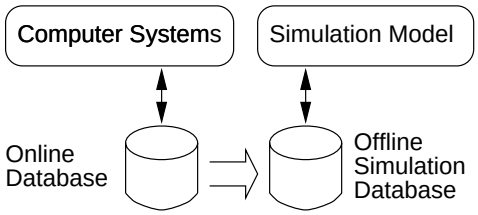


Figure 14.1: *The database based input method based on a copy of the ERP database.*

Maintenance The simulation model maintenance effort should be limited.

Usability Data used to drive the simulation should be generated and maintained in software and through interfaces the users are used to. User interfaces for production plan inputs and database modifications should be simple for fast generation of new experiments.

Simulation speed Each simulation should be executed and analyzed within two hours, say, making several iterations with model modifications possible within one day.

The following sections will present the method and how these requirements have been achieved.

14.3 Model Building

The simulation model is built automatically from the resource register. Adding additional fields for geometry and location in the resource register will allow automatic generation of the visual model as well. Maintaining a geometry register is a rather limited effort since changes are small and seldom made. However, as the model is large and complex, visual interpretation of the simulation is hardly practical.

The model objects share common logic to retrieve the information required to run the model. The input data collection is reduced to identifying the correct ERP database tables and copy them to the off-line simulation model database, as shown in Figure 14.1. Furthermore, the model logic can be simplified considerably since the operation lists in the database has information about routings, which generally has to be embedded in the simulation model. The result is thus a completely database-driven simulation model.

14.4 Used Software

The modeling and simulation tool used was QUEST from DELMIA. The project was managed using CVS (Cederqvist 1993) for configuration management.

Documentation of the project and the model utilized previously developed methods for integrated documentation presented in Chapter 11.

The DBMS used, MySQL, is a fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL is free software and is licensed under the GNU General Public License (MySQL www). Using SQL enables portability between different DBMSs. A SQL DBMS being available on a number of platforms was chosen for portability reasons since the developed tools were to be used on both SGI IRIX and MS Windows platforms.

14.5 Developed Software

The database integration was performed through a set of modules. The QDBC (Quest DataBase Connection) module contained server and client routines to connect the QUEST model to the database. The SQL module was used to format queries. The third module contained the routines to execute a model driven by a database using the former two modules. A fourth module that was to break down the master plan, using ordinary MRP I (Manufacturing Requirements Planning), is yet to be developed.

14.5.1 The QDBC Server

The QDBC server was implemented in C and relied on the MySQL C interface and the socket routines supplied with QUEST. The database, host, user, password and port to communicate through were specified when starting the server. The server read messages on the TCP/IP (Transmission Control Protocol/Internet Protocol) socket and passed the query to the database. The reply was then sent back to the client together with message identification, message size and the number of rows and columns. Using TCP/IP allowed the database and simulation engine to reside on different computers anywhere on a network.

14.5.2 The QDBC Client Routines

The QDBC client routines were implemented in QUEST SCL. The routines read the raw reply from the database and stored it in a dynamic list data structure. A dynamic list structure was used for scalability, i.e. any number of rows, fields and any size of the fields are allowed. A number of routines were supplied to facilitate abstract and simple retrieval of individual rows or fields. The QDBC client routines could be rewritten in C/C++ to enable exchange of the simulator.

14.5.3 The SQL Routines

The SQL routines were supplied to simplify the programming and hide the SQL implementation from the user. Thus, if another DBMS would be used

```
-- Start by removing the existing database.
DROP DATABASE IF EXISTS mrp;

-- Create the database.
CREATE DATABASE IF NOT EXISTS mrp;

-- Use the created database.
USE mrp

-- Create operation list table.
CREATE TABLE IF NOT EXISTS operations (
article_nbr CHAR(12),
resource_nbr CHAR(4),
operation_nbr INT(3),
setup_time REAL,
load_time REAL,
cycle_time REAL,
unload_time REAL);

-- load the data
LOAD DATA LOCAL INFILE "operations.txt"
INTO TABLE operations;
```

Figure 14.2: *An example of the SQL commands used to load the database.*

it was simple to change the SQL syntax in the routines without affecting the simulation model. This set of routines could also be rewritten in C/C++ for portability reasons.

14.6 The Database

The original database was exported as field delimited ASCII files into a simulation database. The simulation database had a defined set of tables that was known to the simulator routines. Thus can the ERP system database come from any vendor since the export/import of input data defines the interface and only minor modifications were expected to be required. The simulation database is created and data is loaded using a few SQL commands, shown in Figure 14.2. In the proof-of-concept model the tables and fields in Table 14.1 were used. The simple database can be said to be a miniature ERP system database.

A work calendar and shifts for all resources in the factory had to be defined as it was not implemented in the ERP system and had to be defined manually.

Table 14.1: *The database tables and fields in the proof-of-concept model.*

Table	*Fields*
orders	order number, article number, number of items, start date, stop date
operations	article number, resource number, operation number, setup time, load time, cycle time, unload time
articles	article number, article name, standard cost
resources	resource number, resource name, setup time, load time, cycle time, unload time, resource cost

14.7 The Simulation Model Objects

14.7.1 The Orders

The parts passing through the simulation model were in this case orders. An order contained the search keys necessary to retrieve information from the database at any point in the simulation.

As an order passed through the system, the article database could be updated, i.e. decreasing an article register by requiring articles and increasing an article register by generating articles.

Orders would be generated from a master plan defined by the user. Each product could be made in a number of variants and historic data was to be used to set the product variant mix. From the master plan and the structure register, products were to be broken down to the article level and production orders would be generated over time. The master plan break down routine in the ERP system had long lead-time and was therefore to be performed in a separate module.

14.7.2 The Scheduler

The scheduler first determined the first start date from the production plan, i.e. the simulation time origin. Starting from the first start date, each date was stepped through day by day. The scheduler retrieved the orders of the current simulation date from the database and then each order was passed to the first element in the operations list. Thus the material flow was started for the entire simulation model using only one element. The dates were generated using the built in date routines in the database server, removing the need to explicitly build a calendar routine. It should be noted that the inter release time of orders can easily be changed, to e.g. an hour or a week, depending on the requirements at the company under study.

```
require order
curr_op = get_current_op(art_nbr)
work res_setup_time()
for ii = 1 to nbr_of_items do
   work res_load_time()
   work res_cycle_time()
   work res_unload_time()
endfor
```

Figure 14.3: *The process logic pseudocode of the resource class.*

```
while (num_routed < num_to_route) do
   next_res = get_next_res(art_nbr, \
                           curr_op)
   route order to next_res
   num_routed = num_routed + 1
endwhile
```

Figure 14.4: *The route logic pseudocode of the resource class.*

14.7.3 The Resources

Each resource is of the same class and has the same logic. The difference between instances of a class is the geometric representation, location, shifts and the name. All element information is stored in the database.

The general resource process logic retrieves the setup, load, cycle and unload times from the database and executes them before passing the order to the next element. The process logic of the resource class is straightforward as shown in the pseudocode in Figure 14.3.

The route logic is general and is used by all elements driven by the database, including the scheduler. The route logic determines the next resource on the operation list and then routes the order, shown in the pseudocode in Figure 14.4.

14.7.4 The Terminating Element

The terminating element is common for the entire simulation model and used to store statistics and update registers before removing the orders from the simulation.

14.8 Running the Simulation

Orders are generated from a master plan and the database connection is opened before the simulation is started. The initialization logic read the data-

base in order to create and connect all elements. It should be noted that there was no model at all to begin with, only a simulation initialization routine that created the model.

The article stock database could be initialized with the current actual stock of material to reflect a probable outcome of the next period, referred to as *intelligent initialization* (Banks et al. 2000). The current manufacturing system state might, however, not be available and a warm up period would then be required. The article stock database can also be set randomly to reflect a general situation.

14.9 Discussion and Conclusions

The case has shown that it is easy to integrate a simulation system with other computer systems. The proof-of-concept model has so far shown that it is feasible to model and simulate entire factories using existing ERP databases and that it can be performed with limited resources. Furthermore, it has been shown that it is relatively simple to generate the software and routines necessary to implement a database driven simulation model.

It is generally recommended to build simulation models with the stochastic behavior correctly modeled (Banks et al. 2000, Law & Kelton 1991). The presented simulation model is dynamic, but has limited stochastic behavior due to the lack of data for failures and cycle time variations in the ERP database. However, the simulation accuracy is expected to be better that a static CRP (Capacity Requirements Planning) analysis, but less than the accuracy achieved with traditional DES methods. This is the trade off when using readily available, but not complete, information. The result is a highly reduced modeling time and an ability to model large factory models. Still, the simulation is dynamic and interactions are captured. If failure statistics are available, they are easily added to the database and thus to the simulation model. However, it has to be considered whether failure losses are included in the aggregated cycle times in the database.

Most of the routines and software is generalizable to any enterprise. The method presented is general in that ERP databases contain the same sets of data although being stored differently in another enterprise. By developing a database export interface the simulation database can be made independent of the original ERP database. Furthermore, the method is also platform and DBMS independent. It is expected to be possible to simulate another enterprise from scratch with limited efforts.

Costs for maintenance of the simulation model is anticipated to be reduced considerably compared to traditional methods of building simulation models since the simulation data is updated automatically as the ERP database is updated in the daily operations.

As pointed out by Iuliano (1995) different versions of the used software tools will create compatibility problems over time, not only in between ver-

sions of the same software tool but also in between different integrated software tools. Due to the modular design these problems will be reduced although not removed.

The simulation model can be designed to handle several different performance measures. Furthermore, the performance measure sampling is easily implemented in all the objects in the simulation model by changing a few general routines. Delivery date, wait times, lead-times and WIP were planned to be used as performance measures. Adding output data filters will facilitate fast and easy identification of possible problems in the factory from the large amount of output data.

The developed methods for integration of existing ERP databases can also be used for manufacturing system redesign or modifying control strategies. Delayed deliveries will work as a measure of how well a production plan can be fulfilled. Wait times indicate bottlenecks. The differences in between delayed and early deliveries in combination with wait times can be used to trace, not only, what orders are delayed, but also why. The larger the difference between delayed and early deliveries the more reason to modify control strategies. A large number of both early deliveries and delayed deliveries imply that there are synchronization problems, i.e. the early deliveries have been processed at the expense of the delayed deliveries. The simulation model can thus be used to increase the order promise accuracy, maximize utilization of production capacity and to modify and test control strategies before implementation.

The simulation database is a copy of the real database not to affect the daily operations. By running the simulation off-line the database can be updated from the simulation and modified to simulate how, e.g. an investment, would affect performance measures without affecting the daily operations.

So far only the basic routines to build and run the simulation model have been implemented. However, the work remaining to get a fully operational factory simulator is limited. The work remaining is also context dependent and has therefore been left out until a real case study can be performed.

Since the proof-of-concept tools have not been tested in a full scale, there are a couple of questions to be considered. Firstly, the system was planned to be used during one day with a few iterations. This puts a demand on the simulation speed. Since it is a large system that is to be simulated with a large number of events it is unknown how fast simulation results could be generated. Secondly, methods to extract vital data from the vast amounts of output data have to be developed. It is this data that is the basis for making decisions on both for the next simulation iteration and finally for the next master plan. Thirdly, the accuracy of the simulation results has to be verified. The input data is deterministic and might not be up to date.

In summary it can be said that the usability issues probably does not pose any problems. Technically it is quite simple to filter data from a database and once the right parameters have been determined this will pose no problem. The accuracy, however, has to be verified in a real case.

Part V

Epilogue

Chapter 15

Discussion

This chapter starts with a comparison between CAD, RS, and DES to put the following discussion in a context. Then conceived problems with DES are discussed. The following section discusses the contributions presented and how they address the conceived problems. Finally there is a discussion on the validity of the contributions.

15.1 Comparison Between CAD, RS and DES

The current usage of CAD, RS and DES is compared and it is considered for how long the tools have been used. The comparison is made to study how usable each tool has been considered to be in an industrial context. It is considered for how long each tool have been around for use by researchers and specialists and for how long the tools have been around for mainstream use. The time comparisons are by no means exact, but give and indication of the speed of implementation of the tools from early systems to mainstream use.

The first CAD systems were developed in the late sixties and the first working commercial systems were available in the seventies. During the eighties CAD reached widespread industrial usage. RS were available as commercial systems in the mid eighties and widespread usage can be said to have started in the mid or late nineties. DES has been around for about forty years and commercial systems made for widespread use can be said to have been around since the mid eighties. It is stated that DES still has not reached widespread acceptance by the industry the way CAD and RS have.

The widespread usage of the tools currently differs fundamentally, which is argued to be related to the benefits using the tools conceived by the industry compared to the conceived cost of using the tools.

In the case of CAD it is very easy to show that it is more feasible to make modifications to a CAD model than with eraser and pen on paper, i.e. the main advantage of the tool is how easy it is to make changes. Current tools also

facilitate the modeling process.

Similarly, RS has shown that many costly mistakes can be avoided by using the tool. Since the user sees the problems on screen there is no doubt that there are problems and possible solutions can be tested. The cost of modifying the tools, jigs etc. are well known in this case, based on experiences.

DES is used to assess complex systems. As such, there are several parameters affecting performance, both soft and hard. Due to the many parameters and that only a handful of them are modeled, it is far harder to show that DES is valuable. The results have to be interpreted as well, while no interpretation is required in the case of e.g. RS. Furthermore, in the case of DES, there is usually little information to make comparisons to, i.e. the real systems performance, before and after implementation of simulation results, are not known. With CAD and RS one can compare what would be the result with or without the tool and the tools are accepted to the level where alternatives are not even considered while that is not the case for DES. With the exception of a few cases, there are few hard core proofs presented that show that tangible economic benefits have been achieved when using DES.

With this comparison it is concluded that there has to be one or several problems still associated with DES that has to be solved before it is widely accepted by the industry for mainstream use.

15.2 Problems

15.2.1 Cost and Benefits

There is a limited usage of DES in the Swedish industry. Possible causes are lack of knowledge, failure to see the benefits compared to the cost, and the complexity of the usage of the technology. The knowledge issue can not be addressed by a thesis and instead the focus has been on facilitating the usage of DES.

I have been using DES in several projects and have experienced some of the advantages and problems. The main advantage is then the possibility to make fairly correct model of a complex system and then study the system over time and thereby gain both qualitative and quantitative information about the system. The main problem is then the conceived benefit compared to the conceived cost of using DES (an excellent discussion on this topic can be found in Holst & Bolmsjö 2002). I have not addressed the problem of the conceived benefit of using the tool directly. Indirectly the conceived benefit problem has been addressed by presenting a methodology that generates credible simulation results which is one part of the problem. Note that the word *conceived* is used on purpose as explained below.

In the cases performed the benefits of the simulation studies were seldom assessed or even been asked for. The customers believed in DES to start with, and that is probably why no evaluation was made.

It is believed that DES benefits, and especially economical benefits, has to be highlighted to a larger extent than has been done traditionally. In the next few paragraphs I will elaborate on the problem of quantifying the benefits of DES using the presented case studies as examples.

In the Profilgruppen case study an assessment of the quantitative benefits was made and the simulation study was shown to be profitable by avoiding cost.

In the first BT case study the conceived bottleneck was the robot cell and therefore there were plans in investing in another cell. The simulation showed that there was another bottleneck in the system and that investing in another robot cell would not yield higher throughput and thus that cost was avoided. There was also a set of qualitative benefits in the study. The maximum throughput in three of the main cells in the system could be determined. It was thus known to what extent it was worth investing to increase throughput in one of the cells before another cell would limit throughput. The cost avoided with this kind of knowledge can not be quantified unless the cost of such investments already are assessed without knowing the maximum required throughput, i.e. simulation results should not be used to be able to correctly calculate the difference between the investment made if the simulation results were *not* known and the investment made if the simulation results were known. Such strategies can of course not be employed just to assess possible cost reductions on the behalf of DES.

In the second BT case study such reduced cost assessments was difficult to make since no changes of the developed manufacturing system was required to achieve the objectives and no cost reductions could be proved. Was the simulation without benefits then? One could argue that if the simulation study would not have been used the investment risk would be higher. How is risk quantified unless there are probabilistic chances of quantified outcomes of different, in this case nonexistent, scenarios? On the other hand, if there would have been a performance problem in the system and it could be solved in the simulation study such and assessment could have been done, but again, that was a nonexistent scenario.

The integrated factory simulation was developed to assess the impact of master plans on resource utilization and delayed deliveries. There is no doubt that the tool has benefits. How can the value of such a simulation tool be evaluated? What is the cost of delayed deliveries? When does delayed deliveries cause customers to change supplier?

The tools developed for quick and correct modeling of the sub-assembly systems at VCBC tries to increase the benefits by reducing modeling time and increasing accuracy. The reduced modeling time is quantifiable, but how can cost reductions based on more accurate simulation models be assessed. Obviously there is a benefit in making correct assessments in a way that the manufacturing system is not too costly and at the same time reach performance objectives. The same problem remains, it is difficult, not to say, impossible, to make assessments of cost reductions. The same holds for the data collection

framework.

Recently interviews with case study contacts was performed to find out what their experiences were considering whether they were about to use DES in future project or not and the reasons for their decision.

According to Michael Admyre at BT Products, they had quantified the reduced cost of using both RS and DES and were confident that there was a benefit of using the tools. They considered the benefits versus cost to be higher for RS than for DES.

Erling Samuelsson at ABB Body-in-White considered the lack of accurate input data such a big problem that there was no reason to continue the use of DES until the problem was solved due to the high accuracy required. Hence the proposed data collection framework.

Joakim Karlborg at Volvo Articulated Haulers believed in simulation in general and there was no doubt they were going to continue to use both RS and DES. No quantitative assessments have been made to my knowledge, the decision seems to be based on good experiences.

Ulf Hansén, formerly at Urshults Werkstads AB, agreed in that there were several benefits, but that DES mainly was suitable for larger projects where the large time consumption could be motivated by large possible cost reductions.

In summary it is generally difficult to prove the benefits of DES. However, decisions more complex than choosing to use or not to use DES is made everyday. These decisions are more based on hunches about what is right or wrong than a detailed quantitative analysis. That seems to be the case for those companies that has chosen to walk the simulation track as well.

15.2.2 A Defense of Discrete-Event Simulation

The following discussion might appear as a fool's defense of a lost cause, but is intended to highlight some issues that are not always considered.

Time and Simulation

Without changing methodology or using different tools, time consumption can be reduced by letting the simulation study life cycle coincide with the manufacturing system life cycle. DES is far to often used as a *fire-and-forget* analysis tool, i.e. for single shot analysis of systems. Fire-and-forget yields low efficiency since the results of the initial large effort is used only once. As an analogy, it is like making a geometrical model of e.g. a car hood for the visual appearance of the car. Then a new model is made from scratch to design the tools and jigs, another one to make crash tests and yet another to make an aerodynamic analysis. In reality the geometrical model is of course reused. How come DES models are not? There is no claim that there is a problem free and seamless reuse of the models in either case.

According to Kusiak & Lee (1996) as much as 60–80 percent of the total manufacturing cost might be affected by the decisions made in the early phases

of product design. There is reason to believe that this holds for the manufacturing system design as well. Therefore DES should be used when developing the manufacturing system, and not to resolve the problems later because of a poor manufacturing system design. It should be noted that there is no guarantee that the manufacturing system design becomes better because DES is used, but the risk is believed to be reduced.

Another reason for the widespread misconception that simulation studies consume more time than they really do, is that the simulation study is viewed as an activity on it's own and not as an effort integrated in the manufacturing system development process or manufacturing system redesign process. No matter how the analysis is performed, a large amount of information has to be collected, but for some reason that entire cost is related to DES and not to the study as a whole. The view put forward here is in line with ABC (Activity Based Costing) and ABM (Activity Based Management) (Ax & Ask 1995, Kaplan & Cooper 1998) where costs are associated with their proper cost drivers.

The presented methodology was mainly designed for usage as an integrated part of the manufacturing system development process where the decision on using DES or other means to assess the system is taken after the conceptional phase, as shown in Figure 10.3.

To achieve a certain quality of the analysis of a manufacturing system the analyst requires a certain amount of information about the studied system. The functional and conceptional phases have to be performed either way. The added effort is then in the design and realization phases. As mentioned, the effort in the design phase is limited while it should be admitted that the realization phase could consume much time, which is one of the reasons for reusing the model. However, the time consumption should then be related to the quality of the analysis. An analogy would be to compare a detailed FEM analysis with a simple calculation made on a calculator. Of course the FEM analysis takes more time, but at the same time, the analysis is of much higher quality. Similarly, one can not compare a simple analysis made with a spreadsheet with the analysis made with a DES tool. Of course one has to consider whether a detailed analysis is required, which is why the decision on whether using DES or not is taken after the conceptual phase.

Complexity and Simulation

Another reason believed to affect the low usage is the conceived complexity of using DES. The complexity issue can be divided into two parts, *tool complexity* and *inherent complexity*.

Most DES tools of today are complex (with a few exceptions as discussed by Johansson et al. 2002), but compared to e.g. CAD or RS tools, there is no significant difference. A user is able to learn the tool basics within a few days, while it takes months to become well acquainted and productive, which holds for all three categories of tools.

The inherent complexity is due to the fact that complex systems are studied

and thus is the analysis of the system complex. Therefore it is stated that it is not the DES tool that is considered complex, instead it is the difficulty making a valid conceptual model that can be translated into a simulation model. The translation process that is quite complex can, however, be related to the tool. The main problem is to make valid simplifications of the studied system that can be used together with the input data that is possible to collect, i.e. there are several difficult decisions to be made and it is hard to know if the chosen paths are good enough.

15.3 Contributions

The first step to reduce time consumption is, based on the discussion in the previous section, to use DES the right way, i.e. not for single shot analysis unless it is believed that the value of the study is greater than the cost of performing it.

The next step in improving the usage of the DES technology is to perform the study with an efficient process, which has been presented in Chapters 10 and 11. The complexity problem has also been addressed by supplying a methodology that structures the simulation study effort and at the same time reduces the time consumption and lead-time of the simulation study. The methodology also tries to overcome the credibility and validity problems by thorough audits performed by the parties in the simulation study. Credibility as well as validity would suffer considerably if the customer and domain experts did not contribute actively in the process.

Input data collection and analysis is identified as a problem by several authors. In Chapters 12 and 14 two approaches have been employed to relax the problem.

In Chapter 12 a framework was presented to collect input data from automated manufacturing systems in a way that very accurate simulation models could be built. Since there would be little motive developing input data collection system for DES studies alone, the input data collected has several other usages as well. The main advantage of the data collection framework is that dependencies otherwise found in input data were avoided. The lack of dependencies and the ability to collect data for devices make generalization of the distribution functions possible which in turn make possible the assessment of new, not yet existent, manufacturing systems. As pointed out, such generalizations are complicated due to the dependencies not addressed by the framework, but far more accurate than if no or little data is available.

In Chapter 14 generally available input data was used to relax the input data problem and at the same time speed up the modeling process considerably. The trade off was that the information content was limited, which resulted in a model less accurate than desired. The approach did, however, result in an extremely fast model building process and the maintenance of the model was reduced considerably. The approach is also extremely scalable in that there is

no relationship in between the size of the simulation model and the modeling effort. The presented approach enables assessment of the impact of different master plans and is believed to be applicable to almost any enterprise utilizing MRP I for their production planning.

The other approach employed to reduce the modeling effort was to generalize the modeling components so that parameterized models could be used, as presented in Chapter 13. The approach, as many other similar approaches, was limited in that only manufacturing systems of the studied type could be modeled.

In Chapter 12 tools for accurate assessment of the size of safety zones and the number of operators in the manufacturing system was presented. The data collection framework, parameterized models and safety zone tools together forms an approach that will make possible quick modeling of accurate DES models for manufacturing systems of the type studied.

15.4 Construct Validity

The presented contributions aim at reducing time requirements for performing simulation studies in general and in particular contexts. However, proving that it is so is quite difficult.

The approaches presented in Chapters 13 and 14 will obviously reduce time consumption in the contexts they are designed to work.

To prove that the methodology presented in Chapters 10 and 11 will reduce time consumption is complicated. There is no practical possibility to perform the same case studies with another methodology and trace the differences. Theoretically there are too many parameters affecting the outcome and in practice it is hardly doable due to scarce resources. What I can refer to is my, and the participants, image of the methodologies applicability. In the second case study performed at BT there was a requirement from BT that SCM should be used which indicates the practical usefulness of that part of the methodology. In the software industry it is an accepted fact that CMM and SCM is required for stable and repeatable development.

15.5 Internal Validity

Internal validity concerns the causal relationships, i.e. whether it was the proposed methodology that generated the reduced time consumption or if was an effect of other variables.

One variable that did affect the differences in time consumption was how trained the participants in the performed case studies were. For instance, the first case study at BT took considerable time due to poor training. In the second case study at BT most parties had a common notion of how to perform the simulation study, thus were there less meetings, less communication, the case was performed without interrupts, etc. That is, all the lessons learned in

the first case was implemented and thus reduced time consumption. The point is here is not to compare the two simulation studies, only to highlight the big differences in performance due to one single parameter although performed within the same methodological framework.

Although it is difficult to pinpoint the causal relationships one can say that the methodology does institutionalize the simulation process and thereby reduces the number of parameters that can affect the simulation process and the impact of the remaining parameters. One of the points in using a methodology is to reduce the variance of the outcome.

It can also be established that concurrent work will reduce lead-time if the amount of communication required in between team members is kept limited. The modular decomposition of the simulation model in the design phase enabled concurrent model building in the realization phase.

The methodology's focus on continuous audits will reduce the number of mistakes made and the impact of those mistakes. Similarly, it can be established that moving time consuming activities forward in the process will reduce the collection of redundant information that otherwise might be collected. These two properties alone will reduce the time consumption of the simulation study.

The structured approach in developing a conceptual model and then designing the simulation model will result in a more structured model and a better understanding of the studied system before the time consuming activities begin. These statements are hard to prove quantitatively with the case studies, but is, for instance, how the software industry works, based on many years of practical experience.

15.6 External Validity

External validity concerns whether the findings are generalizable beyond the immediate case study. As discussed above, there are several parameters affecting the findings and typically several case studies are required.

The methodology presented in Chapter 10 has been used to different extents in the BT Products case studies and in the most recent case study performed at Volvo Articulated Haulers (not presented in this thesis). The methodology was not developed when the Profilgruppen case study was performed and the other projects have been performed to develop methods and tools and are therefore not suitable for comparisons. The experiences from the case studies are that the simulation study process does benefit from the structured approach.

The data collection framework has not been tested in a practical application. However, the input data from the input data collection system at Scania had many of the properties of the proposed input data collection framework. Simulations performed by Arne Ingemansson, Division of Robotics, Department of Mechanical Engineering, Lund University, showed that the accuracy

of the simulation model was good. The other properties follow from logical deduction.

The integrated factory simulation method has not been tested in full scale, but has been validated with a proof-of-concept model. The additional work required to perform a full scale test is limited.

What would be interesting to find out in a real life application is the simulation execution time for such large models and the accuracy of the output. The test database was quite small and thus did not test simulation speed. There was no real factory to compare to and thus the accuracy could not be tested either. There was no guarantee that data for comparisons were available at BT even if a full-scale test would be made. Therefore it was doubtful whether it was possible to make good estimations of accuracy. It should be noted that this is not a new idea, a similar system is used at Volvo Articulated Haulers and the usability of the application is said to be good.

The parameterized modeling method is designed to work in a specific context and is thus not generalizable. However, the general routines that are the basis for the method, are generalizable to any manufacturing system where the objects communicate before performing an action. Several automated manufacturing systems work this way and it is recommended to replicate the *wait-until-condition-satisfied* behavior in such systems. The default behavior in DES systems does not take such behaviors in consideration.

Many of the developed tools were made for usage with QUEST and are thus not directly generalizable, but the principles and ideas are.

15.7 Reliability

The objective of this section is to show that if the same case studies were to be performed the same conclusions would be drawn. One prerequisite for allowing another investigator to repeat an earlier case study is the need to document the procedures followed in the earlier case. A good guideline for doing case studies is therefore to conduct the research so that an auditor could repeat the procedure and arrive at the same results. Since the presented thesis is such a document this requirement is fulfilled to the extent possible.

15.8 Applicability

Since it was stated that the presented research was considered to be applied a short note on that issue is in place. The case studies have been performed together with the industry for which the presented results are aimed. The case studies have been performed together with the industries on their conditions and based on their requirements.

The proposed methodology is general and can be applied to almost any DES study in any industry. The supporting documentation tools are, as discussed,

applicable and desired in principle, but not with the tools currently used. The applicability is also limited to source code, and not to non-editable model files.

The data collection framework and modeling tools developed in the case studies at VCBC were designed to be applied at that company. The data collection framework is applicable to any automated or semi-automated manufacturing system.

The integrated factory simulation was developed for usage at BT and was considered applicable by BT. VAH (Volvo Articulated Haulers) considers the similar system at VAH applicable.

Chapter 16

Conclusions

The overall objective of the presented work is to reduce time consumption and lead-time.

In Chapter 10 the scope has been the discrete-event simulation study process. The simulation study engineering and management processes were described and a model for capability maturity was supplied.

In Chapter 11 a tool for documentation were presented to support the developed simulation study process.

In Chapter 12 a framework for reducing the data collection effort has been presented. The main advantage with the framework is that the collected data is accurate and has few or no dependencies, i.e. data is of high quality. The data collected with a high level of detail that enables the use of the data for real-time operations and manufacturing system development within several domains.

To reduce the development time during in the realization phase two methods have been developed, presented in Chapters 13 and 14, each reducing the development time in two different domains with two different approaches. In Chapter 13 a set of tools was presented that made possible parameterized modeling which in turn enabled the modeling process. In Chapter 14 existent MRP I data was copied to a simulation model database. The method reduced the modeling and maintenance effort considerably.

Another implicit objective has been to increase the DES model credibility.

In Chapter 10 model credibility was enhanced by thorough continuous reviews of the activities performed and the activities to be performed. The capability maturity model also provided a model for continuous improvement of the simulation engineering and management process as well as the simulation organization, i.e. the infrastructure required for stable and reliable execution of simulation studies.

In Chapter 12 model accuracy has been improved by addressing the input data quality. Correct modeling of manufacturing systems of the type studied at VCBC has been addressed in Chapters 12 and 13.

Another conclusion was that to yield high efficiency performing simulation studies requires correct usage, i.e. the simulation study should not be used to correct mistakes made earlier in the manufacturing systems life cycle. The mistakes considered are those not detected due to the omitted usage of DES earlier in the life cycle. Instead the life cycle of the simulation study should coincide with the life cycle of the manufacturing system. Related to this is the widespread misconception that simulation studies consume more time than it actually does. One reason for this misconception is the usage of DES for fire-and-forget simulation studies which follows from the non-coinciding life cycles. It was also concluded that it does take longer time to perform a simulation study compared to simpler methods, but that added effort should be related to the benefits of a more detailed analysis.

Chapter 17

Future Research

The thesis is on integration of DES in the manufacturing system development process and a few problems have been identified. Below a few issues are highlighted that is believed to facilitate the integration of DES into the manufacturing system development process.

First it should be made clear that if DES is to be integrated, it is in fact a number of processes with tools, methods, and methodologies that should be integrated. There is no intention to identify all the issues. That is a research issue on it's own. However, to identify the issues of integration (with all facets) isolated to the usage of DES would be a first step towards an understanding of the complexity of the integration issue. An identification of the information required in the tools and processes in the manufacturing system development process would clarify the requirements and form a base for future research. Examples are SFCS development, robot cell development, and manufacturing system process development. The mentioned domains are all related to the simulation of the entire manufacturing system. Work has been performed in these areas, although more efforts are required and, above all, standards for the exchange and sharing of the manufacturing system design information base are required.

The reason for integration of the information base is to reduce time consumption and cost. In the case of DES the integration of information would reduce the modeling effort and the input data problem.

Similarly to the integration of the information used in the manufacturing system development it is believed that integration of e.g. SFCSs with DES has the possibility to increase simulation accuracy, reduce development time consumption, and reduce start up times. Integration of applications would in a DES context reduce the modeling time.

This thesis has to some extent treated the integration of processes, i.e. the DES process with the manufacturing system development process. However, it is believed that more research is required in this area to make this integration mature.

Bibliography

Abrahamsson, T. & Törnblad, H. (1998), Design och analys av layout med diskret händelsestyrd simulering vid UWAB, Bachelor thesis, Högskolan i Växjö, Institutionen för teknik och naturvetenskap. (In Swedish).

Ahlmann, H. & Hagberg, L. (190/91), Tekniska och ekonomiska aspekter på driftsäkerhet och underhåll, Institutionen för Industriell Organisation, LTH, via (Ericsson 1997).

Al-Dabass, D. & Cheng, R., eds (1999), *UKSIM'99. Fourth National Conference of the UK Simulation Society*, Nottingham Trent Univ.

Andersson, N. (1997), Volvo PV Göteborg, QUEST satsning, Presented by Stefan Blomgren at Deneb Swedish User Conference 1997, Gothenburg.

Andradóttir, S., Healy, K. J., Withers, D. H. & Nelson, B. L., eds (1997), *Proceedings of the 1997 Winter Simulation Conference*.

ARGESIM (www), 'Arbeitsgemeinschaft simulation: Comparisons'. Available online from `http://ws3.atv.tuwien.ac.at/comparisons/`, [accessed April 24 2002].

Asklund, U. (1999), Configuration Management for Distributed Development — Practice and Needs, PhD thesis, Department of Computer Science, Lund University.

Asklund, U. & Magnusson, B. (1997), A case-study of configuration management with ClearCase in an industrial environment, *in* R. Conradi, ed., 'Software Configuration Management. ICSE '97 SCM-7 Workshop Proceedings', Springer-Verlag, pp. 201–21.

Ax, C. & Ask, U. (1995), *Cost Management: Produktkalkylering och ekonomistyrning under utveckling*, Studentlitteratur. (In Swedish).

Babich, W. A. (1986), *Software Configuration Management*, Addison-Wesley, Reading, Massachusetts.

Balci, O. (1998), Verification, validation, and accreditation, *in* Benjamin et al. (1998), pp. 41–8.

Ball, P. D. (1995*a*), Embedding simulation into the process of designing manufacturing systems., *in* 'Proceedings of the Eleventh National Conference on Manufacturing Research', De Montfort University, Leicester, UK, pp. 403–7.

Ball, P. D. (1995*b*), Tailoring a generic, object-oriented simulation system to model manufacturing systems, *in* 'Proceedings of the 1st DYCOMANS Workshop : Industrial Control and Management Methods: Theory and Practice', Prague, Czech Republic, pp. 1–5.

Ball, P. D. (1996), Introduction to discrete event simulation., *in* 'Proceedings of the 2nd DYCOMANS workshop on Management and Control: Tools in Action', Algarve, Portugal, pp. 367–76.

Ball, P. D. & Love, D. M. (1992), Design of an extensible manufacturing simulator, *in* 'Simulation and AI in Computer Aided Techniques, Proceedings of the European Simulation Symposium', Society for Computer Simulation, Dresden, Germany, pp. 491–5.

Ball, P. D. & Love, D. M. (1994), 'Expanding the capabilities of manufacturing simulators through the application of object-oriented principles.', *Journal of Manufacturing Systems* **13**(6), 412–23.

Ball, P. D., Boughton, N. J. & Love, D. M. (1994), 'Extending the boundaries of manufacturing simulation', *International Journal of Manufacturing System Design* **1**(2), 99–109.

Banks, J., Aviles, E., McLaughlin, J. R. & Yuan, R. C. (1991), 'The simulator: New member of the simulation family', *Interfaces* **21**(2), 78–86. via (Ball & Love 1994).

Banks, J., Carson, J. S. & Nelson, B. L. (1996), *Discrete-event system simulation*, 2nd edn, Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458.

Banks, J., Carson, J. S., Nelson, B. L. & Nicol, D. (2000), *Discrete-Event System Simulation*, 3rd edn, Prentice-Hall, Inc., Upper Saddle River, New Jersey 07458.

Banks, J., ed. (1998), *Handbook of Simulation — Principles, Methodology, Advances, Applications, and Practice*, John Wiley & Sons, Inc.

Bartolotta, A., McLean, C., Lee, Y. T. & Jones, A. (1998), Production systems engineering: Requirements analysis for discrete-event simulation, Technical Report NISTIR 6154, National Institute of Standards and Technology. Available online from `http://www.nist.gov/`, [accessed June 2 2001].

Baudouin, M., Ruberti, C., Arékion, J. & Kieffer, J.-P. (1995), A decision support system based on a factory wide information integrated system and discrete event simulation to help solve scheduling problems in a

semiconductor manufacturing environment, *in* 'Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95', IEEE Comput. Soc. Press, pp. 437–45 vol.2.

Becker, H. S. (1958), 'Problems of inference and proof in participant observation', *American Sociological Review* **23**(7), 652–60. via (Yin 1994).

Benjamin, P., Erraguntla, M. & Mayer, D. D. R., eds (1998), *Proceedings of the 1998 Winter Simulation Conference*, IEEE.

Benneyan, J. C. (1998), 'Software review: Stat::Fit', *OR/MS Today*. Available online from `http://www.lionhrtpub.com/orms/orms-2-98/swr.html`, [accessed May 12 2000].

Bernard, B. (2000), Integration of Manufacturing Simulation Tools with Information Sources, Licentiate Thesis, Computer Systems for Design and Manufacturing Division, Department of Manufacturing Systems, Royal Institute of Technology, Stockholm, Sweden.

Bersoff, E. H., Henderson, V. D. & Siegel, S. G. (1980), *Software Configuration Management: An Investment in Product Integrity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Bley, H. & Wuttke, C. C. (1999), Multiple use of simulation models for production systems, *in* Brussel, Kruth & Lauwers (1999), pp. 379–88.

Bobeanu, C. & Filip, F.-G. (1995), ModCPN - an integrated set of tools for Petri nets-based modelling of manufacturing systems, *in* 'Proceedings 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation. ETFA'95', Vol. 1, IEEE Comput. Soc. Press, pp. 95–103.

Bolier, D. & Eliën, A. (www), 'SIM : a C++ library for discrete event simulation'. Available online from `http://www.cs.vu.nl/vakgroepen/se/eliens/sim/sim_html/sim.html`, [accessed October 13 1999].

Bolmsjö, G. & Gustafsson, B. (1998), Concurrent development of product and manufacturing technologies using advanced simulation methods, *in* N. Mårtensson, R. Mackay & S. Björgvinsson, eds, 'Changing the ways we work: Shaping the ICT-solutions for the next century: Proceedings of the Conference on Integration in Manufacturing', Vol. 8 of *Advances in design and manufacturing*, IOS Press, pp. 432–40.

Bolmsjö, G., Lorentzon, U. & Randell, L. (1999), Integration of development of product and production system using advanced simulation tools, *in* Brussel et al. (1999), pp. 389–98.

Boughton, N. J. (1995), Modelling manufacturing planning and control systems: The application of object-oriented principles and discrete-event simulation, PhD Dissertation, The University of Aston in Birmingham. Thesis number DX193955.

Bridge, K. (1990), The application of computerised modelling techniques in manufacturing system design., PhD Thesis, Aston University, Birmingham U.K.

Brunnermeier, S. B. & Martin, S. A. (1999), Interoperability cost analysis of the U.S. automotive supply chain, Final Report RTI Project Number 7007-03, Research Triangle Institute, Center for Economics Research, Research Triangle Park, NC 27709. Available online from `http://www.rti.org/publications/cer/7007-3-auto.pdf`.

Brussel, H. V., Kruth, J.-P. & Lauwers, B., eds (1999), *Proceedings of The 32nd CIRP International Seminar on Manufacturing Systems — New Supporting Tools for Designing Products and Production Systems*, Katholieke Universiteit Leuven - Departement Werktuigkunde.

Carrie, A. (1988), *Simulation of Manufacturing Systems*, John Wiley & Sons Ltd.

Case study evaluations (1990), U.S. General Accounting Office, Program evaluation and Methodology Division, Washington D.C.: Government Printing Office. Available online from `http://www.gao.gov/special.pubs/10_1_9.pdf`, [accessed March 18 2002].

Cederqvist, P. (1993), *Version Management with CVS: for CVS 1.10*, Signum Support AB, Box 2044, 580 02 Linköping, Sweden.

Centeno, M. A. & Standridge, C. R. (1993), Databases: designing and developing integrated simulation modeling environments, *in* Evans, Mollaghasemi, Russell & Biles (1993), pp. 526–34.

Chalmers, A. F. (1999), *What is this thing called Science?*, 3rd edn, Hacket Publishing Company Inc.

Charnes, J. M., Morrice, D. J., Brunner, D. T. & Swain, J. J., eds (1996), *Proceedings of the 1996 Winter Simulation Conference*, SCS Int.

Chew, J. & Sullivan, C. (2000), Verification, validation, and accreditation in the life cycle of models and simulations, *in* Joines & Barton (2000), pp. 813–8.

Collins, N. & Watson, C. M. (1993), Introduction to Arena, *in* Evans et al. (1993), pp. 205–12. via (Ball 1995*a*).

Conwell, C. L., Enright, R. & Stutzman, M. A. (2000), Capability maturity models support of modeling and simulation verification, validation, and accreditation, *in* Joines & Barton (2000), pp. 819–28.

Crnkovic, I., Persson Dahlqvist, A. & Svensson, D. (2001), Managing complex systems — challenges for PDM and SCM, *in* 'IEEE Asia-Pacific Conference on Quality Software', Hong Kong.

Dart, S. A. (1990), Spectrum of functionality in configuration management systems, Technical Report CMU/SEI-90-TR-11 ESD-90-TR-212, Carnegie Mellon, Software Engineering Institute. Available online from `http://www.sei.cmu.edu/legacy/scm/tech_rep/TR11_90/TOC_TR11_90.html`, [accessed October 16 2000].

Dart, S. A. (1992*a*), Parallels in computer-aided design framework and software development environment efforts, *in* 'IFIP Transactions A (Computer Science and Technology)', Vol. A-16, pp. 175–89.

Dart, S. A. (1992*b*), The past, present, and future of configuration management, Technical Report CMU/SEI-92-TR-8 ESC-TR-92–8, Carnegie Mellon, Software Engineering Institute. Available online from `http://www.sei.cmu.edu/legacy/scm/abstracts/abscm_past_pres_future_TR08%_92.html`, [accessed October 16 2000].

De Smet, R., Gelders, L. & Pintelon, L. (1997), 'Case studies on disturbance registration for continuous improvement', *Journal of Quality in Maintenance Engineering* **3**(2), 91–108.

Delen, D., Benjamin, P. C. & Erraguntla, M. (1998), Integrated modeling and analysis generator environment (IMAGE): a decision support tool, *in* Benjamin et al. (1998), pp. 1401–8.

Delen, D., Benjamin, P. C. & Erraguntla, M. (1999), An integrated toolkit for enterprise modeling and analysis, *in* Farrington et al. (1999), pp. 289–97.

Drake, G. R. & Smith, J. S. (1996), Simulation system for real-time planning, scheduling, and control, *in* Charnes, Morrice, Brunner & Swain (1996), pp. 1083–90.

Dufrene, D., Gharbi, A., Kieffer, J. P., Rebouche, J. Y. & Villeneuve, L. (1994), A discrete event simulation methodology for the design of flexible manufacturing systems: toward the use of an integrated set of models following progressive detail levels, *in* D. K. Pace & A.-M. Fayek, eds, 'Proceedings of the 1994 Summer Computer Simulation Conference. Twenty-Sixth Annual Summer Computer Simulation Conference', SCS, pp. 561–6.

Eatock, J., Serrano, A., Giaglis, G. M. & Paul, R. J. (1999), A case study on integrating business and network simulation for business process redesign, *in* Al-Dabass & Cheng (1999), pp. 114–18.

Ericsson, J. (1997), Störningsanalys av tillverkningssystem: Ett viktigt verktyg inom Lean Production, PhD thesis, Lund University, Department of Production and Materials Engineering. CODEN: LUTMDN/TMMV-1034)/1-227(1997) (in Swedish).

Eriksson, U. (1997), Discrete Event Simulation Methodical Output Analysis, Licentiate Thesis, Luleå University of Technology, Department of Material

and Manufacturing Engineering, Division of Manufacturing Enginering. ISSN: 1402-1757, ISRN:LTU-LIC-1997/21-SE.

Eriksson, U. (1999), Production Engineering, Chalmers University of Technology, SE 412 96 Göteborg, Sweden. Working paper.

Evans, G. W., Mollaghasemi, M., Russell, E. C. & Biles, W. E., eds (1993), *Proceedings of the 1993 Winter Simulation Conference*, IEEE.

Eversheim, W., Bochtler, W., Grassler, R. & Kolscheid, W. (1997), 'Simultaneous engineering approach to an integrated design and process planning', *European Journal of Operational Research* **100**(2), 327–3. via (Johansson 2001).

ExpertFit (www), 'ExpertFit: Distribution-fitting software'. Available online from `http://www.averill-law.com`, [accessed May 12 2000].

Farrington, P. A., Black Nembhard, H., Sturrock, D. T. & Evans, G. W., eds (1999), *Proceedings of the 1999 Winter Simulation Conference*, IEEE, Phoenix, Arizona, USA.

Feldman, S. I. (1979), 'Make — a program for maintaining computer programs', *Software — Practice and Experience* **9**(4), 255–65.

Fishwick, P. A. & Cubert, R. M. (www), 'SimPack'. Available online from `http://www.cis.ufl.edu/~fishwick/simpack/simpack.html`, [accessed October 13 1999].

Freedman, S. (1999), An overview of fully integrated digital manufacturing technology, *in* Farrington et al. (1999), pp. 281–5.

Fujimoto, R. M. (1999), Parallel and distributed simulation, *in* Farrington et al. (1999), pp. 122–31.

Giaglis, G. M. (1999), Integrated simulation of business processes and information systems, *in* Al-Dabass & Cheng (1999), pp. 119–25.

Giaglis, G. M., Paul, R. J. & O'Keefe, R. M. (1999), 'Research note: integrating business and network simulation models for IT investment evaluation', *Logistics Information Management* **12**(1-2), 108–17.

Gmilkowsky, P., Eckardt, F. & Palleduhn, D. (1997), Concept of an integrated system for modeling, simulation and analysis of discrete production processes, *in* M. S. Obaidat, ed., 'Proceedings of the 1997 Summer Computer Simulation Conference Simulation and Modeling Technology for the Twenty-First Century', SCSI, pp. 329–34.

Gmilkowsky, P., Eckardt, F. & Palleduhn, D. (1998), Automated simulation model generation: a knowledge-based approach within an integrated system for modeling, simulation and analysis of discrete production processes, *in* R. Ades, M.; Griebenow, ed., 'Proceedings of the Simulators International XV', Society for Computer Simulation, pp. 157–62.

gpdes3d (www), 'dpdes3d'. Available online from `http://sourceforge.net/projects/gpdes3d`, [accessed June 5 2002].

Grewal, N. S., Bruska, A. C., Wulf, T. M. & Robinson, J. K. (1998), Integrating targeted cycle-time reduction into the capital planning process, *in* Benjamin et al. (1998), pp. 1005–10 vol.2.

Gullander, P. & Klingstam, P. (1998), Survey on development, operation and maintenance of manufacturing systems, Technical Report PTE 98:02, Department of Production Engineering, Chalmers. via (Holst et al. 2000*a*).

Gustafsson, B. (1998), 'Om att ge ökad spridning åt VSOP-tekniken', VSOP-konferens III, Institutet för Verkstadsteknisk Forskning. (In Swedish).

Hacking, I. (1983), *Representing and intervening: Introductory topics in the philosophy of natural science*, Cambridge Univ. Press.

Harrell, C. & Tumay, K. (1995), *Simulation Made Easy: A Manager's Guide*, Engineering & Management Press.

Heitmann, K., Hirschberg, A., Rauh, E. & Wunderlich, J. (1997), Zusammenfassung der Studie: Einsatz der Simulationstechnik, Technical report, Institut für Werkzeugmaschinen und Betriebswissenschaften (iwb) der Technischen Universität München und Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik (FAPS) der Universität Erlangen-Nürnberg (in German).

Hirschberg, A. G. & Heitmann, K. (1997), Simulation in German industry - a survey, *in* W. Hahn & A. Lehmann, eds, 'Simulation in Industry - 9th European Simulation Symposium 1997, Passau, Germany', Society for Computer Simulation International, pp. 429–33.

Hitchens, M. (1996), 'Software modeling: Simulation, emulation and monitoring', *APICS — The Performance Advantage* pp. 62–6.

Hitchens, M. W. (1989), Direct connect emulation and the project life cycle, *in* MacNair, Musselman & Heidelberger (1989), pp. 843–7.

Hlupic, V., Irani, Z. & Paul, R. J. (1999), 'Evaluation framework for simulation software', *International Journal of Advanced Manufacturing Technology* **15**(5), 366–82.

Hoffman, D. R. (1998), An overview of concurrent engineering, *in* 'Annual Reliability and Maintainability Symposium 1998 Proceedings. International Symposium on Product Quality and Integrity', IEEE, pp. 1–7.

Holst, L. & Bolmsjö, G. (2001), 'Simulation integration in manufacturing system development: A study of Japanese industry', *Industrial Management & Data Systems* **101**(7), 339–56.

Holst, L. & Bolmsjö, G. (2002), Activity-based management applied to discrete-event simulation projects: A conceptual framework, in *Proceedings of The 35th CIRP International Seminar on Manufacturing Systems* (*Proceedings of The 35th CIRP International Seminar on Manufacturing Systems* 2002), pp. 134–41.

Holst, L., Randell, L. & Bolmsjö, G. (2000*a*), Integrated development of manufacturing systems using simulation — Proposing the fundamentals for a joint research project, *in* 'Proceedings of The 33rd CIRP International Seminar on Manufacturing Systems', Stockholm.

Holst, L., Randell, L. & Bolmsjö, G. (2000*b*), The integrated use of simulation in the development of manufacturing systems: A study of Japanese industry, *in* INF (2000), pp. 1475–82.

Ilar, T. (1997), A Method for Supporting Implementation of Semi-Automated Manufacturing Cells, Licentiate Thesis, Luleå University of Technology, Department of Materials and Manufacturing Engineering, Division of Manufacturing Enginering. ISSN: 1402-1757, ISRN: LTU-LIC-1997/20-SE.

INF (2000), *Proceedings of INFORMS-KORMS Seoul 2000 Conference*.

Ingemansson, A. (2001), Reduction of Disturbances in Manufacturing Systems Based on Discrete-Event Simulation, Licentiate in Engineering Thesis, Department of Mechanical Engineering, Lund University.

Ingemansson, A. (2002), Iakttagelser kring linje D16 med hjälp av händelsestyrd simulering — ett försök att identifiera störningar med hjälp av datainsamlingssystemet, Internal report, Division of Robotics, Department of Mechanical Engineering, Lund University. (In Swedish).

Ingemansson, A. & Bolmsjö, G. (2001), Increase the total output when disturbances are reduced in a manufacturing system, *in* 'International CIRP Design Seminar — Design in the New Economy', International Institution for Production Research, Stockholm, Sweden.

Iuliano, M. (1995), Overview of the manufacturing engineering toolkit prototype, Technical Report NISTIR 5730, National Institute of Standards and Technology. Available online from `http://www.nist.gov/msidlibrary/doc/iuliano.ps`, [accessed June 8 2001].

Iuliano, M. (1997), The role of product data management in the manufacturing engineering toolkit, Technical Report NISTIR 6042, National Institute of Standards and Technology. Available online from `http://www.nist.gov/msidlibrary/doc/ir6042.pdf`, [accessed June 8 2001].

Iuliano, M. & Jones, A. (1996), Controlling activities in a virtual manufacturing cell, *in* Charnes et al. (1996), pp. 1062–7.

Iuliano, M., Jones, A. & Feng, S. (1997), An analysis of AP213 for usage as a process plan exchange format, Technical Report NISTIR 5992, National Institute of Standards and Technology. Available online from `http://www.nist.gov/msidlibrary/doc/ir5992.pdf`, [accessed June 8 2001].

Jackson, M. (1998), Discrete Event Simulation — A Tool to Increase Agility in Production Systems, Licentiate Thesis, Linköping University, Sweden, Division of Assembly Technology, Department of Mechanical Engineering. LiU-Tek-Lic-1998:56.

Järneteg, B. (1995), Time Based Transformation of Manufacturing Systems, PhD thesis, Assembly Technology, Department of Mechanical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Johansson, B., Johnsson, J. & Eriksson, U. (2002), An evaluation of discrete event simulation software for "dynamic rough-cut analysis", in *Proceedings of The 35th CIRP International Seminar on Manufacturing Systems* (*Proceedings of The 35th CIRP International Seminar on Manufacturing Systems 2002)*, pp. 306–13.

Johansson, M. (2001), Information Management for Manufacturing System Development, PhD thesis, Division of Computer Systems for Design and Manufacturing, Department of Production Engineering, Royal Institute of Technology, SE-100 44 Stockholm, Sweden.

Johansson, M. & Rosén, J. (1998), Product realisation and virtual manufacturing based on standard communication, *in* 'Proceedings of SIG-PM Produktmodeller -98', pp. 493–507. via (Johansson & Rosén 1999).

Johansson, M. & Rosén, J. (1999), Presenting a core model for a virtual manufacturing framework, *in* Brussel et al. (1999), pp. 417–23.

Joines, J. & Barton, R., eds (2000), *Proceedings of the 2000 Winter Simulation Conference*, IEEE, Orlando, Florida, USA.

Judd, R. P. & Abell, J. (1996), Off-line development of factory control code using simulation, *in* 'Proc. of the 1996 Summer Computer Simulation Conference', pp. 92–6.

Kaplan, R. S. & Cooper, R. (1998), *Cost & Effect: Using Integrated Cost Systems to Drive Profitability and Performance*, Harvard Business School Press, Boston, Massachusetts.

Kemmerer, S. J., ed. (1999), *STEP: The Grand Experience*, National Institute of Standards and Technology, Gaithersburg, MD. NIST Special Publication 939.

Kiessling, R. & Sandén, G. (1981), Underhållsteknik, Technical report, Sveriges Mekanförbund. via (Ericsson 1997).

Klingstam, P. (1999), A methodology for supporting manufacturing system development: Success factors for integrating simulation in the engineering process, *in* Brussel et al. (1999), pp. 359–67.

Klingstam, P. & Gullander, P. (1997), Overview of simulation tools for computer-aided production engineering, *in* '1997 Advanced Summer Institute - ASI'97', Budapest, Hungary.

Kosturiak, J. & Gregor, M. (1995), 'Simulation von produktionssystemen, teil 2', *Zeitschrift fur Wirtschaftlichen Fabrikbetrieb* **90**(3), 120–4. via (Klingstam & Gullander 1997).

Krause, F.-L. & Jansen, H. (1999), New supporting tools for designing products, *in* Brussel et al. (1999), pp. 1–21.

Kubiak, M. & Tsambasopoulos, C. (1998), Konceptstudie av jigless welding för ram A 40, Bachelor thesis, Högskolan i Växjö, Institutionen för teknik och naturvetenskap. (In Swedish).

Kuivanen, R. (1996), 'Disturbance control in flexible manufacturing', *International Journal of Human Factors in Manufacturing* **6**(1), 41–56.

Kusiak, A. & Lee, G. H. (1996), Design for manufacturing environment, *in* 'Proceedings of the 1996 IEEE IECON. 22nd International Conference on Industrial Electronics, Control, and Instrumentation', Vol. 1, IEEE, pp. XLV–LVI.

Law, A. M. & Kelton, D. W. (1991), *Simulation Modeling and Analysis*, Industrial Engineering and Management Science, 2nd edn, McGraw-Hill Inc.

Lee, Y. T. (1999*a*), Information modeling: From design to implementation, *in* S. Hahavandi & M. Saadat, eds, 'Proceedings of the Second World Manufacturing Congress, WMC'99', International Computer Science Conventions, ICSC Academic Press, Durham, United Kingdom, pp. 315–21.

Lee, Y. T. (1999*b*), An overview of information modeling for manufacturing systems integration, Technical Report NISTIR 6382, National Institute of Standards and Technology.

Lie, C. H., Hwang, C. L. & Tillman, F. A. (1977), 'Availability of maintained systems: A state-of-the-art survey', *AIIE Transactions*.

Lindgren, G. & Rychlik, I. (1997), *Tillförlitlighet och säkerhet: Statistiska metoder och tekniker*, 3rd edn, Lunds Universitet och Lunds Tekniska Högskola, Institutionen för Matematisk Statistik. (In Swedish).

Liyanage, K. & Perera, T. (1998), Design and development of a rapid data collection methodology, *in* 'International Conference on Simulation '98', IEE, pp. 297–304.

Love, D. & Barton, J. (1996), 'Evaluation of design decisions through CIM and simulation', *Integrated-Manufacturing-Systems* **7**(4), 3–11.

Love, D. M. & Bridge, K. (1988), Specification of a computer simulator to support the manufacturing system design process, *in* '3rd International Conference on Computer–Aided Production Engineering', Ann Arbor, Michigan, USA, pp. 317–23. via (Ball & Love 1994).

Love, D. M., Clarke, S. R. & Gooden, D. I. (1987), The use of simulation to determine operational policies in an MRP/FMS environment, *in* J. A. McGeough, ed., 'Second International Conference on Computer-Aided Production Engineering', Mech. Eng. Publications, Bury St. Edmunds, UK, pp. 261–6.

MacNair, E. A., Musselman, K. J. & Heidelberger, P., eds (1989), *Proceedings of the 1989 Winter Simulation Conference*, SCS.

Magnusson, B., Asklund, U. & Minör, S. (1993), 'Fine-grained revision control for collaborative software development', *SIGSOFT Software Engineering Notes* **18**(15), 33–41.

McClanahan, K. (1996), 'Track That Code', BYTE, McGraw-Hill.

McHaney, R. (1988), Bridging the gap: Transferring logic from a simulation into an actual system controller, *in* '1988 Winter Simulation Conference Proceedings', IEEE, pp. 583–90.

McHaney, R. & White, D. (1998), 'Discrete event simulation software selection: An empirical framework', *Simulation & Gaming*.

McLean, C. (1993), Computer-aided manufacturing system engineering, *in* 'Proceedings of the IFIP TC5/WG5.7 International Conference on Advances in Production Management Systems, APMS '93'.

McLean, C. & Leong, S. (1997), Industrial need: Production system engineering integration standards, Technical Report NISTIR 6019, National Institute of Standards and Technology.

Miles, T. I. (1989), Using discrete-event computer simulation to test control systems, *in* MacNair et al. (1989), pp. 848–58.

Miller, P. (1999), *Aegis: A Project Change Supervisor: Reference Manual*. Version 3.20, Available online from `http://www.canb.auug.org.au/~millerp/aegis/aegis.html`, [accessed January 19 2000].

Minör, S. & Magnusson, B. (1993), A model for semi-(a)synchronous collaborative editing, *in* G. de Michelis, C. Simone & K. Schmidt, eds, 'Proceedings of the Third European Conference on Computer-Supported Cooperative Work, ECSCW '93', Kluwer Academic Publishers, pp. 219–31.

Mittelbach, F., Duchier, D., Braams, J., Woliński, M. & Wooding, M. (1999), *The DocStrip Program*, 2.5b edn.

Moorthy, S. (1999), Integrating the CAD model with dynamic simulation: Simulation data exchange, *in* Farrington et al. (1999), pp. 276–80.

MySQL (www). Available online from `http://www.mysql.com`, [accessed March 22 2001].

Nakajima, S. (1988), *Introduction to TPM — Total Productive Maintenance*, Productivity Press.

Nord, C., Pettersson, B. & Johansson, B. (1998), *TPM, Total Productive Maintenance, med erfarenhet från Volvo*, IVF-skrift 97849. In Swedish.

Norman, M., Tinsley, D., Barksdale, J., Wiersholm, O., Campbell, P. & MacNair, E. (1999), Process and material handling models integration, *in* Farrington et al. (1999), pp. 1262–7 vol.2.

Nwoke, B. & Nelson, D. (1993), 'An overview of computer simulation in manufacturing', *Industrial Engineering* **25**(7), 43–57. via (Klingstam & Gullander 1997).

Olsson, T. (1994), Group awarness using fine-grained revision control, *in* 'Proceedings of NWPER'94, Nordic Workshop on Programming Environment Research', Lund, Sweden.

OMNeT++ (www), 'OMNeT++ — Object-oriented discrete event simulator'. Available online from `http://www.hit.bme.hu/phd/vargaa/omnetpp.htm`, [accessed June 5 2002].

Östman, J. (1998), Virtual reality och virtual prototyping - tillämpningar i amerikansk fordonsindustri, Technical report, Sveriges Tekniska Attachéer. Available online from `http://www.statt.se/extern/rapporter/1998_97/usa9806.pdf`, [accessed June 6 2001], (In Swedish).

Oura, J. (2000), A methodology for information system development encouraging end-user participation concept and application to a Japanese manufacturing industry, *in* INF (2000), pp. 232–9.

Pålsson, K. & Quist, J. (2002), Integration av diskret händelsestyrd simulering i utvecklingsprocessen för produktionssystem — En studie vid Volvo Cars Body Components i Olofström, Master's thesis, Division of Robotics, Department of Mechanical Engineering, Lund University. (In Swedish).

ParaSol (www), 'ParaSol C++ library for parallel discrete event simulation'. Available online from `http://www.cs.purdue.edu/research/PaCS/parasol.html`, [accessed June 5 2002].

Parks, C. M., Koonce, D. A., Rabelo, L. C., Judd, R. P. & Sauter, J. A. (1994), 'Model-based manufacturing integration: A paradigm for virtual manufacturing systems engineering', *Computers and Industrial Engineering* **27**(1-4), 357–60.

Patel, R. & Tebelius, U. (1987), *Grundbok i forskningsmetodik*, Studentlitteratur, Lund. (In Swedish).

Paulk, M. C. (1998), Using the software CMM in small organizations, *in* 'The Joint 1998 Proceedings of the Pacific Northwest Software Quality Conference and the Eighth International Conference on Software Quality', Portland, Oregon, pp. 350–61. Available online from `http://www.sei.cmu.edu/cmm/`, [accessed May 31 2002].

Paulk, M. C., Curtis, B., Chrissis, M. B. & Weber, C. V. (1993*a*), Capability maturity model for software, version 1.1, Technical Report CMU/SEI-93-TR-24, DTIC Number ADA263403, Software Engineering Institute. Available online from `http://www.sei.cmu.edu/cmm/`, [accessed May 31 2002].

Paulk, M. C., Weber, C. V., Garcia, S. M., Chrissis, M. B. & Bush, M. W. (1993*b*), Key practices of the capability maturity model, version 1.1, Technical Report CMU/SEI-93-TR-25, DTIC Number ADA263432, Software Engineering Institute. Available online from `http://www.sei.cmu.edu/cmm/`, [accessed May 31 2002].

PDES (1998), Results of STEP testing in the U.S. automotive industry, Technical report, PDES, Inc. via (Brunnermeier & Martin 1999).

Persson Dahlqvist, A., Asklund, U., Crnkovic, I., Hedin, A., Larsson, M. & Ranby, J. (2001*a*), Product Data Management and Software Configuration Management — Similarities and differences, Technical report, The Association of Swedish Engineering Industries.

Persson Dahlqvist, A., Crnkovic, I. & Larsson, M. (2001*b*), Managing complex systems — Challenges for PDM and SCM, *in* 'Tenth International Workshop on Software Configuration Management'.

Peters, B. A. & Smith, J., eds (2001), *Proceedings of the 2001 Winter Simulation Conference*, Arlington, Virginia, USA.

Peters, B. A., Smith, J. S., Curry, J., LaJimodiere, C. & Drake, G. R. (1996), Advanced tutorial-simulation-based scheduling and control, *in* Charnes et al. (1996), pp. 194–8.

Pidd, M. (1992*a*), *Computer Simulation in Management Science*, 3rd edn, John Wiley & Sons Ltd, Chichester, England.

Pidd, M. (1992*b*), 'Guidelines for the design of data driven generic simulators for specific domains.', *Simulation* **59**(4), 237–43. via (Ball & Love 1994).

Pidd, M. (1996), Five simple principles of modelling, *in* Charnes et al. (1996), pp. 721–8.

Prabhu, V. V. & Duffie, N. A. (1995), A distributed simulation approach for enabling cooperation between entities in heterarchical manufacturing systems, *in* R. Lumia, ed., 'Modeling Simulation, and Control Technologies for Manufacturing', Vol. 2596, SPIE—The International Society for Optical Engineering, pp. 234–42.

Prasad, B. (1996), *Concurrent Engineering Fundamentals*, Vol. I and II, Prentice Hall PTR, New Jersey. via (Starbek et al. 1999).

Pritsker, A. A. B. & O'Reilly, J. J. (1998), AweSim: the integrated simulation system, *in* Benjamin et al. (1998), pp. 249–55 vol.1.

Proceedings of The 35th CIRP International Seminar on Manufacturing Systems (2002), Seoul, Korea.

Randell, L. G. & Bolmsjö, G. S. (2001), Database driven factory simulation: A proof-of-concept demonstrator, *in* Peters & Smith (2001), pp. 977–83.

Randell, L., Holst, L. & Bolmsjö, G. (1999), Incremental system development of large discrete-event simulation models, *in* Farrington et al. (1999), pp. 561–8.

Randell, L., Holst, L. & Bolmsjö, G. (2000), Documenting discrete-event simulation models, *in* INF (2000), pp. 1483–90.

Rooks, B. W. (1997), 'Off-line programming: a success for the automotive industry', *Industrial Robot* **24**(1), 30–4.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. (1991), *Object-Oriented Modeling and Design*, Prentice-Hall, Inc, Englewood Cliffs, New Jersey 07632.

Sargent, R. G. (2000), Verification, validation, and accreditation of simulation models, *in* Joines & Barton (2000), pp. 50–9.

Sargent, R. G. (2001), Some approaches and paradigms for verifying and validating simulation models, *in* Peters & Smith (2001), pp. 106–14.

Savén, B. (1995), Verksamhetsmodeller för beslutstöd och lärande: En studie av diskret produktionssimulering vid Asea/ABB 1968-1993, PhD thesis, Linköping University, Department of Computer and Information Science. (In Swedish).

Shewchuk, J. P. & Chang, T. C. (1991), An approach to object-oriented discrete-event simulation of manufacturing systems., *in* B. L. Nelson, W. D. Kelton & G. M. Clark, eds, 'Proceedings of the 1991 Winter Simulation Conference', IEEE, pp. 302–11.

Smith, J. S. & Peters, B. A. (1998), Simulation as a decision-making tool for real-time control of flexible manufacturing systems, *in* 'Proceedings. 1998 IEEE International Conference on Robotics and Automation', IEEE, pp. 586–90 vol.1.

Smith, J. S., Wysk, R. A., Sturrock, D. T., Ramaswamy, S. E., Smith, G. D. & Joshi, S. B. (1994), Discrete event simulation for shop floor control, *in* J. D. Tew, S. Manivannan, D. A. Sadowski & A. F. Seila, eds, 'Proceedings of the 1994 Winter Simulation Conference', IEEE, pp. 962–9.

Solding, P. (2001), Diskret händelsestyrd simulering vid produktion av vindkrafttorn hos Enercon Windtower Production AB, Master's thesis, Division of Robotics, Department of Mechanical Engineering, Lund University. (In Swedish).

Srinivasan, K. & Jayaraman, S. (1997), Integration of simulation with enterprise models, *in* Andradóttir et al. (1997), pp. 1352–6.

Stalk, Jr., G. & Hout, T. M. (1990), *Competing Against Time: How Time-Based Competition is Reshaping Global Markets*, The free Press, New York.

Starbek, M., Kušar, J. & Jenko, P. (1999), Building a concurrent engineering support information system, *in* Brussel et al. (1999), pp. 173–81.

Stat::Fit (www). Available online from `http://www.geerms.com/`, [accessed May 12 2000].

Stroustrup, B. (1991), *The C++ Programming Language*, 2nd edn, Addison-Wesley.

Strub, M. (1998), 'Get in STEP', *Action Line* pp. 16–8. via (Brunnermeier & Martin 1999).

Swain, J. J. (1997), 'Simulation goes mainstream', *OR/MS Today* **24**(5), 35–7. Available online from `http://www.lionhrtpub.com/orms/orms-10-97/simulation-story.html`, [accessed September 18 2000].

Swain, J. J. (1999), 'Imagine new worlds', *OR/MS Today*. Available online from `http://www.lionhrtpub.com/orms/orms-2-99/sim.html`, [accessed September 18 2000].

Szykman, S., Bochenek, C., Racz, J. W., Sriram, R. D. & Senfaute, J. (2000*a*), 'Design repositories: Next-generation engineering design databases', *IEEE Intelligent Systems and Their Applications*.

Szykman, S., Fenves, S. J., Keirouz, W. & Shooter, S. B. (2001), 'A foundation for interoperability in next-generation product development systems', *Computer-Aided Design* **33**(7), 545–59.

Szykman, S., Racz, J., Bochenek, C. & Sriram, R. D. (2000*b*), 'A web-based system for design artifact modeling', *Design Studies* **21**(2), 145–65.

Tatikonda, M. V. & Stietz, M. K. (1994), 'An integrated methodology for operations analysis of computer integrated manufacturing systems: Application to an FMS for manufacture of medical instruments', *International Journal of Advanced Manufacturing* **9**(4), 245–52.

Thuresson, H. & Husberg, P. (1997), Diskret händelsestyrd simulering vid Volvo Articulated Haulers AB, Bachelor thesis, Högskolan i Växjö, Institutionen för teknik och naturvetenskap. (In Swedish).

Trybula, W. (1994), Building simulation models without data, *in* '1994 IEEE International Conference on Systems, Man, and Cybernetics. Humans, Information and Technology', Vol. 1, IEEE, pp. 209–14.

Ulgen, O. M. & Thomasma, T. (1990), 'SmartSim: An object-oriented simulation program generator for manufacturing systems.', *In. J. of Prod. Res.* **28**(9), 1713–30.

Umeda, S. & Jones, A. (1997), Simulation in Japan: State-of-the-art update, Technical Report NISTIR 6040, National Institute of Standards and Technology, U.S. Department of Commerce, Technology Administration.

Vernadat, F. (1996), *Enterprise Modeling and Integration*, 1st edn, Chapman & Hall.

Vinoski, S. (1997), 'CORBA: Integrating diverse applications within distributed heterogenous environment', *IEEE Communications Magazine* **35**(2), 46–55.

Watt, D. G. (1998), Integrating simulation based scheduling with MES in a semiconductor fab, *in* Benjamin et al. (1998), pp. 1713–15 vol.2.

Weaks, H. L. & Barret, J. D. (1997), A demonstration of the Integrated Supportability Analysis and Cost System (ISACS+), *in* Andradóttir et al. (1997), pp. 624–31.

Whitman, L., Huff, B. & Presley, A. (1997), Structured models and dynamic systems analysis: The integration of the IDEF0/IDEF3 modeling methods and discrete event simulation, *in* Andradóttir et al. (1997), pp. 518–24.

Willemain, T. R. (1994), 'Insights on modeling from a dozen experts', *Operations Research* **42**(2), 213–22.

Willemain, T. R. (1995), 'Model formulation: What experts think about and when', *Operations Research* **43**(6), 916–32.

Winner, R. et al. (1988), The role of concurrent engineering in weapon system acquisition, Technical report, Institute for Defense Analyses Report R-338. via (Hoffman 1998).

Xu, J. & AbouRizk, S. (1999), Product-based model representation for integrating 3D CAD with computer simulation, *in* Farrington et al. (1999), pp. 971–7.

Yin, R. K. (1994), *Case study research: design and methods*, 2nd edn, Sage Publications Inc., Thousand Oaks, California.

Appendix A

Proposed Discrete-Event Simulation Process

The discrete-event simulation process presented in Chapter 10 is here shown with IDEFØ notation. The appendix is supplied to complement the description in Chapter 10.

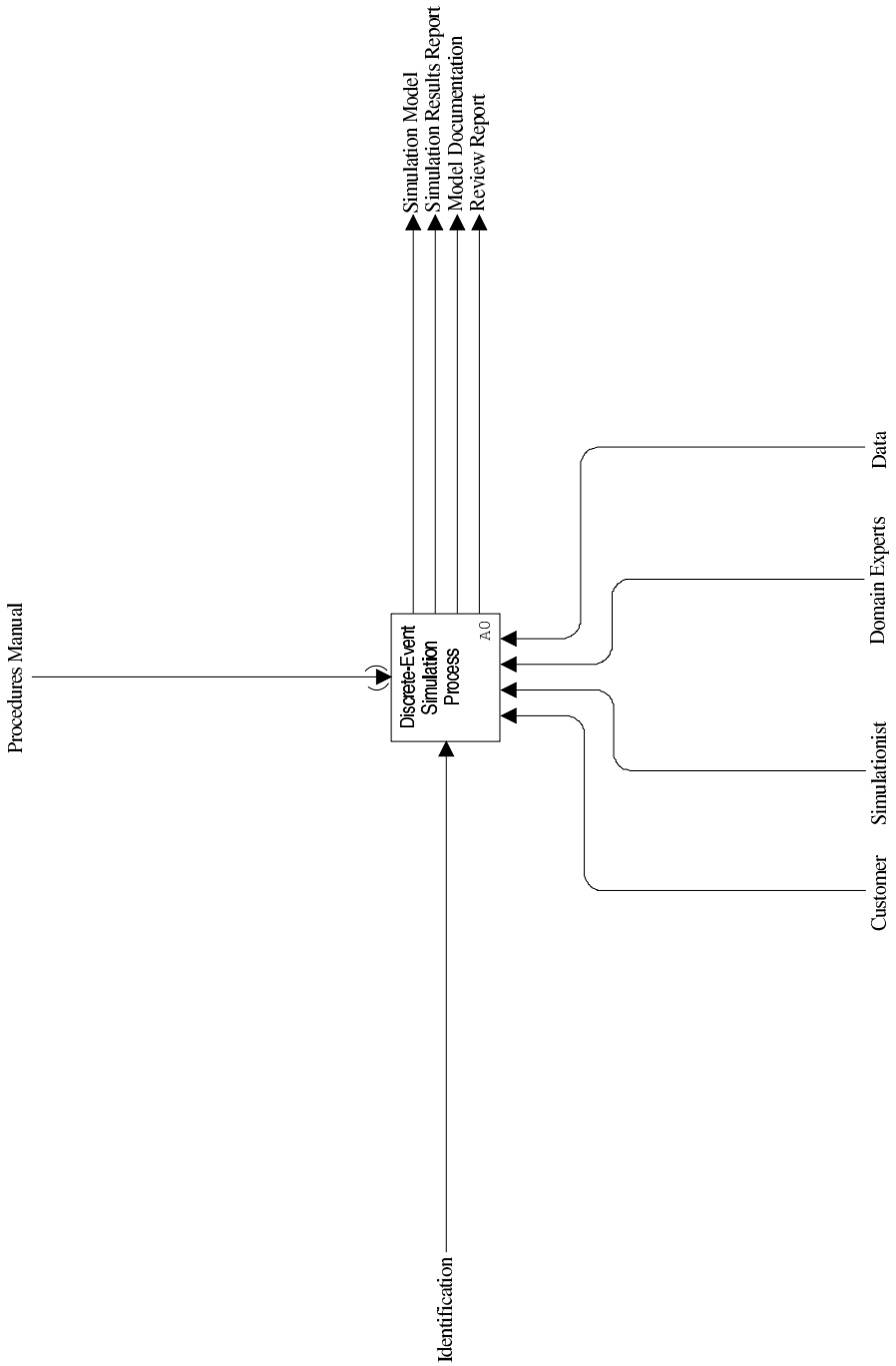


Figure A.1: *Node A-0, Discrete-event simulation process.*

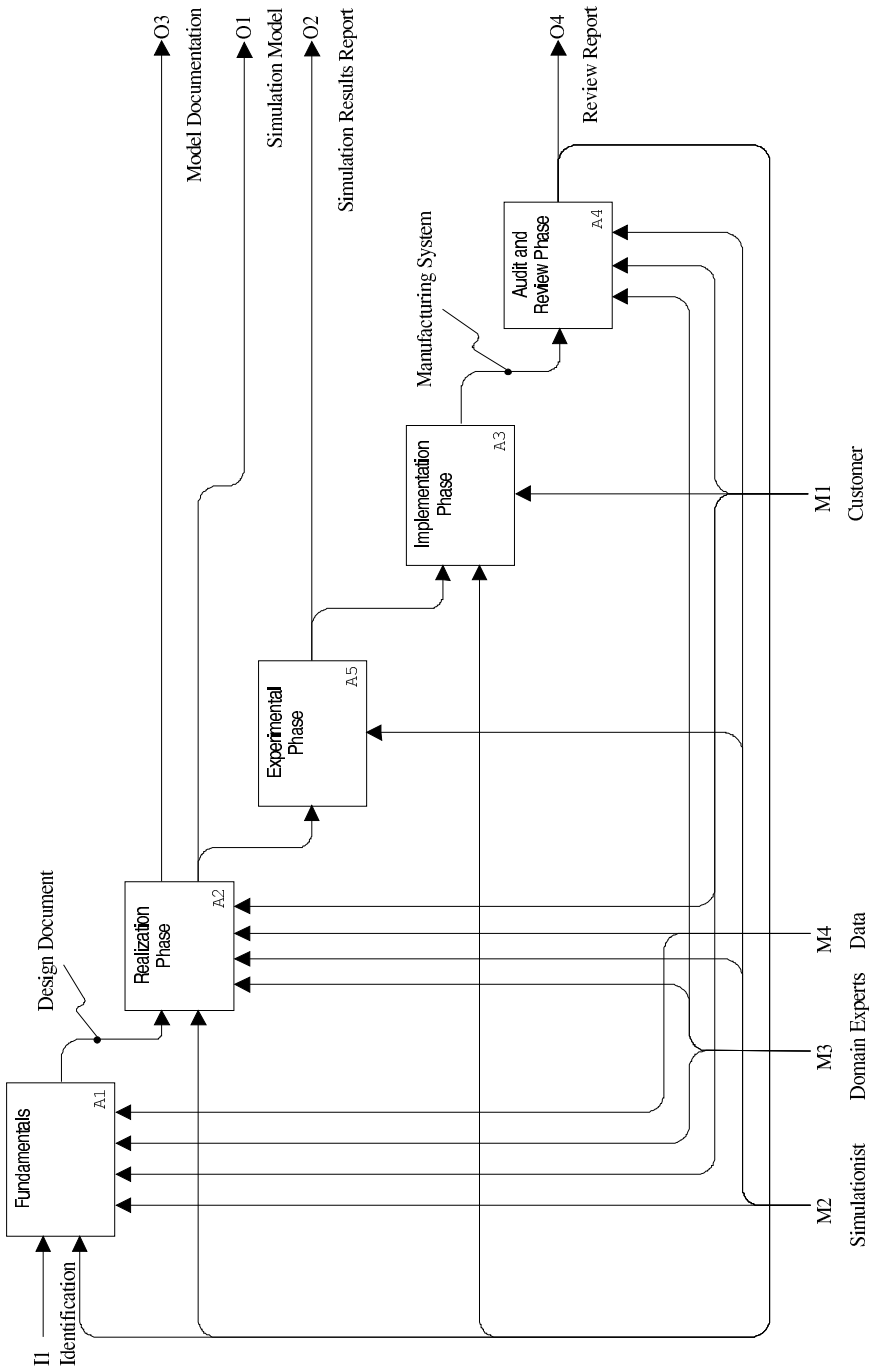


Figure A.2: *Node A0, Discrete-event simulation process.*

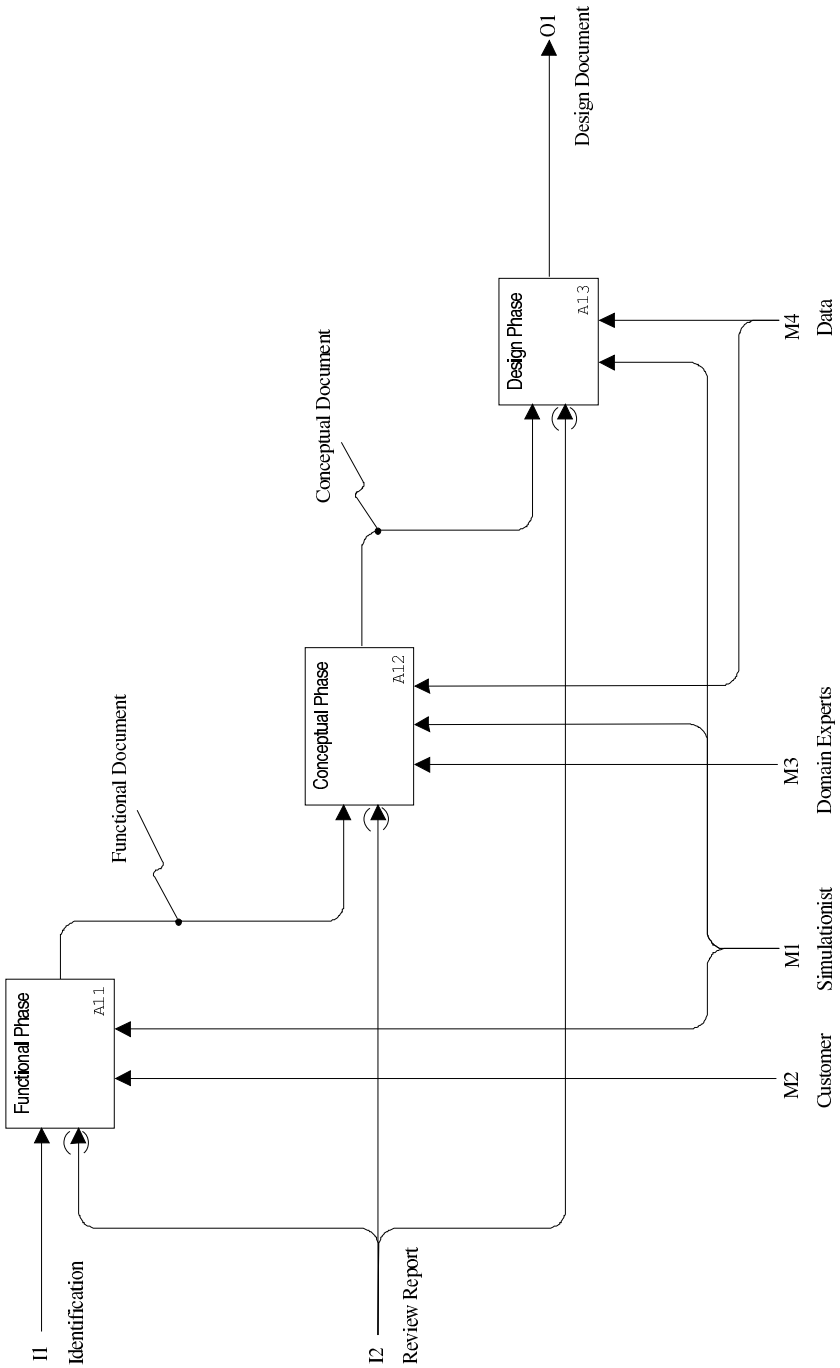


Figure A.3: *Node A1, Fundamentals.*

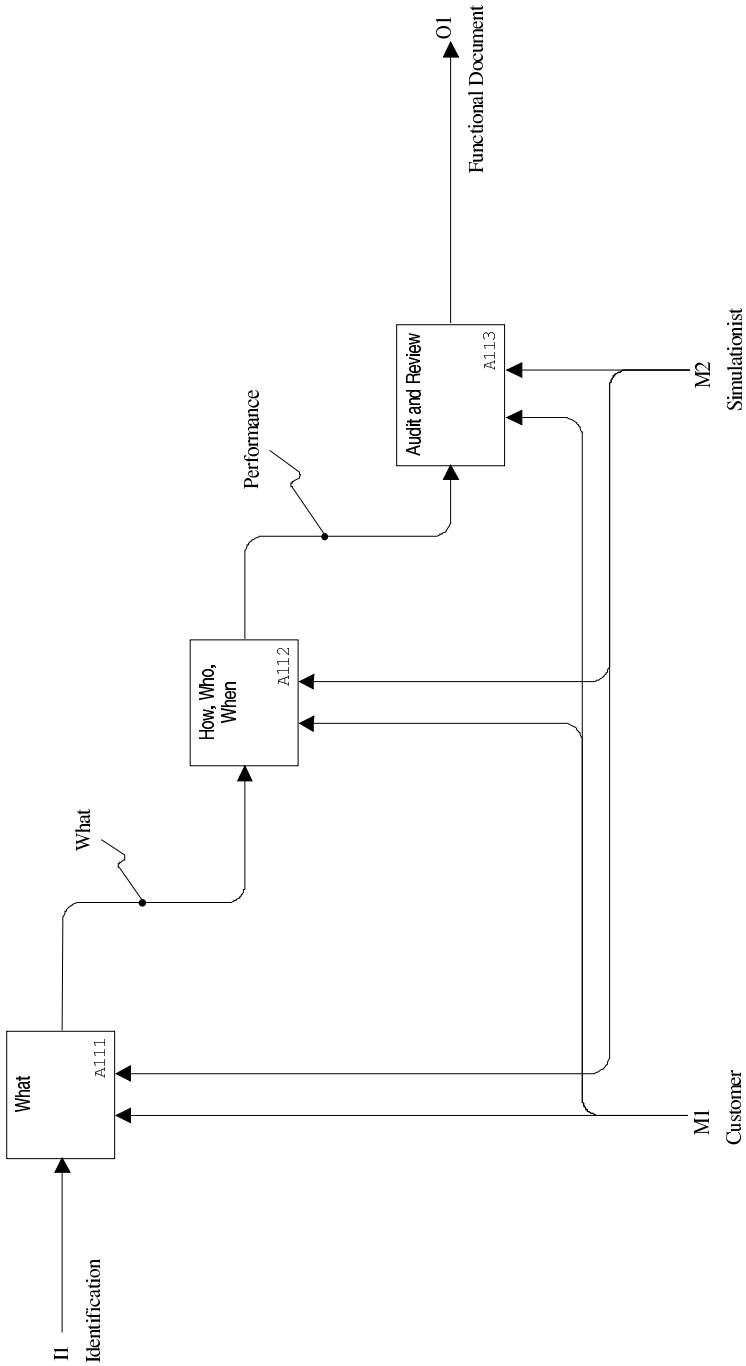


Figure A.4: *Node A11, Functional phase.*

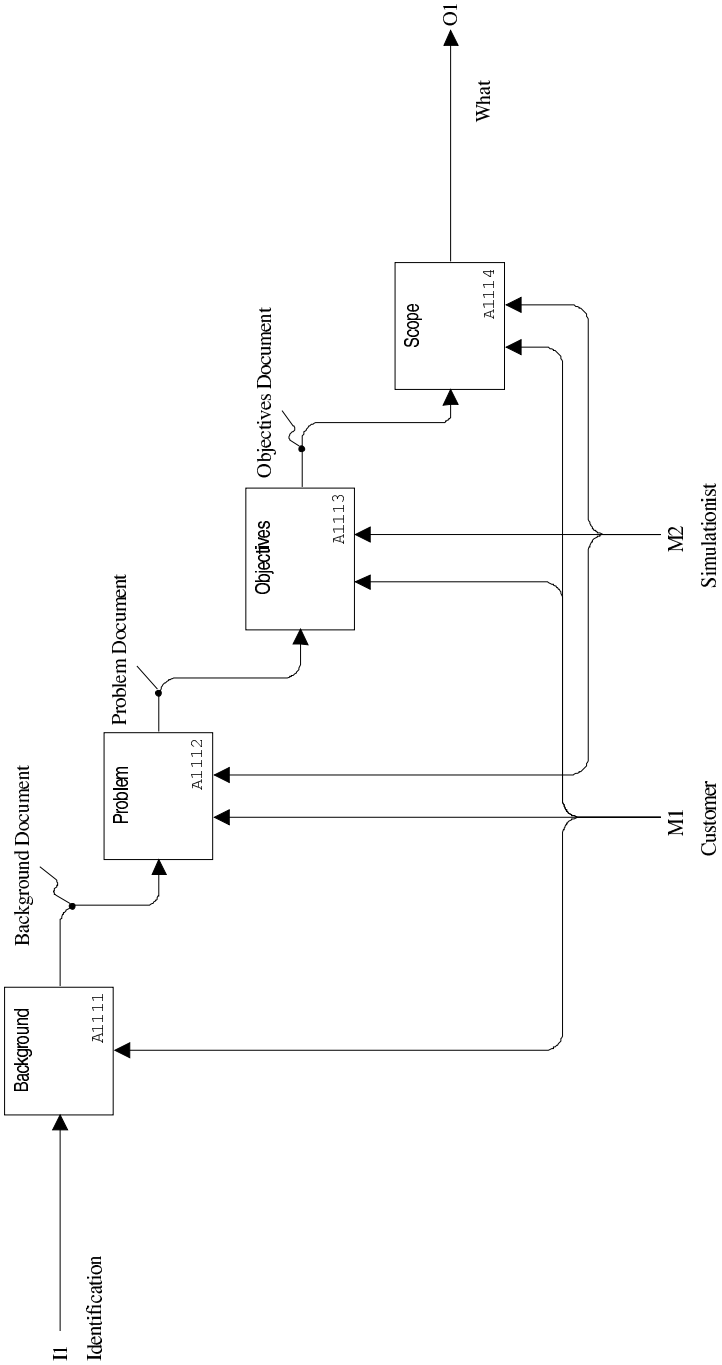


Figure A.5: *Node A111, What.*

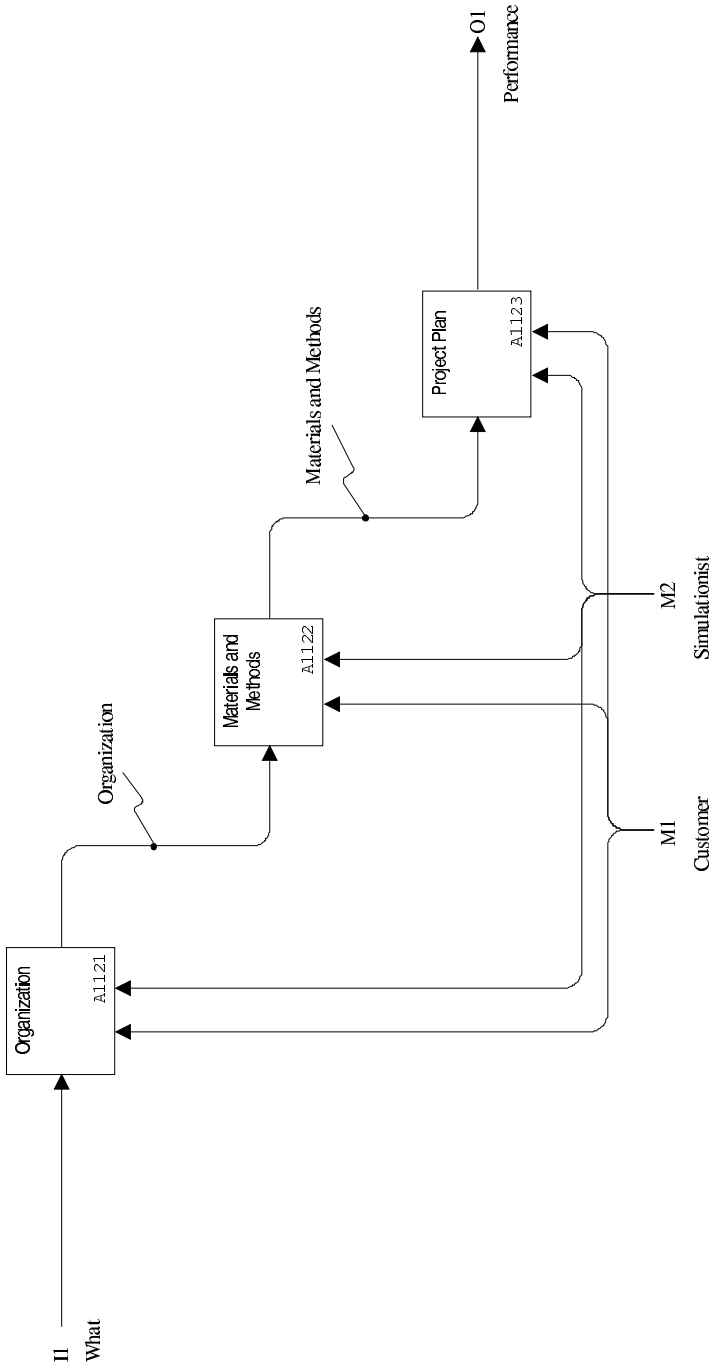


Figure A.6: *Node A112, How, Who, When.*

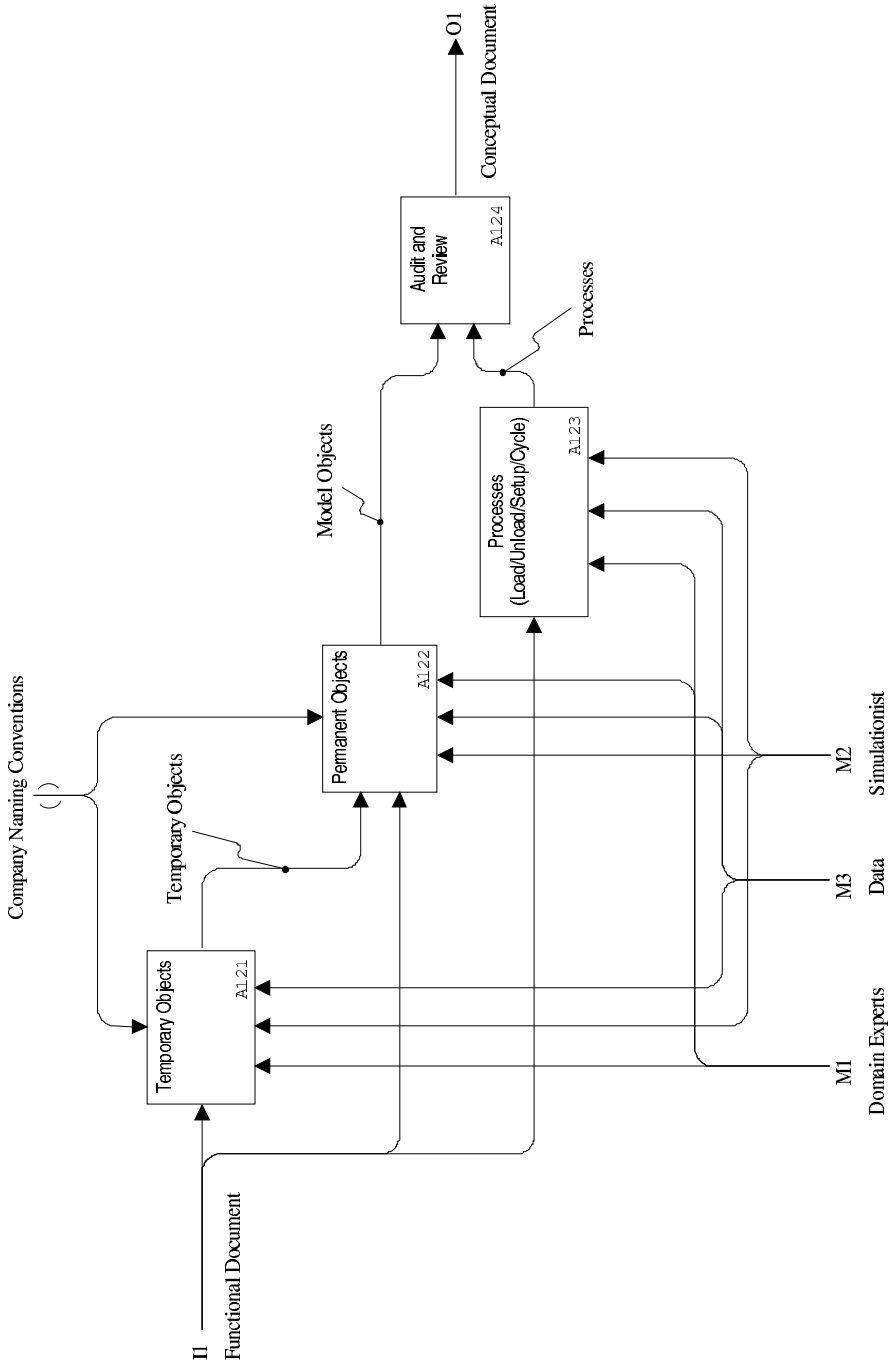


Figure A.7: *Node A12, Conceptual Phase.*

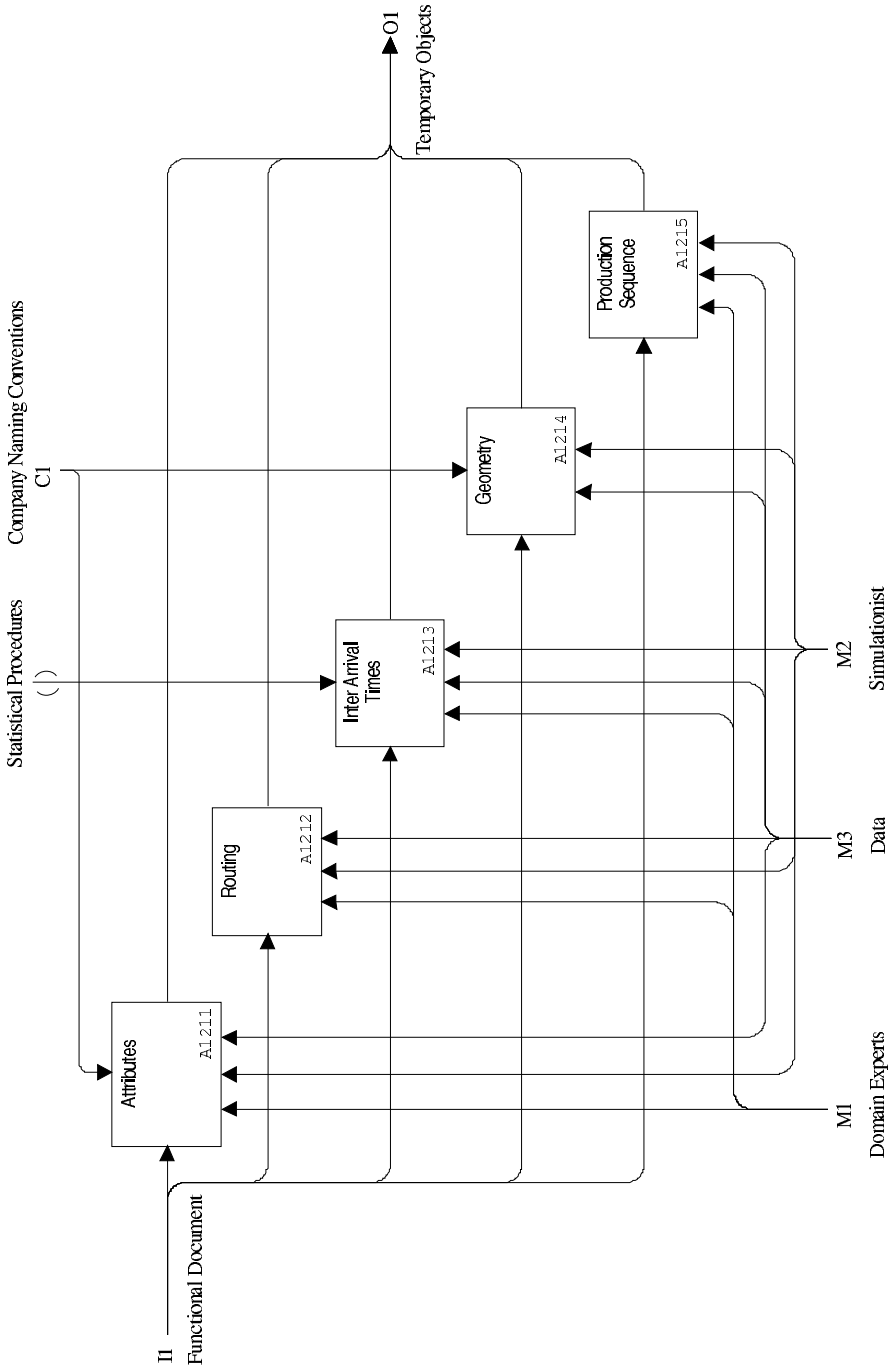


Figure A.8: *Node A121, Temporary Objects.*

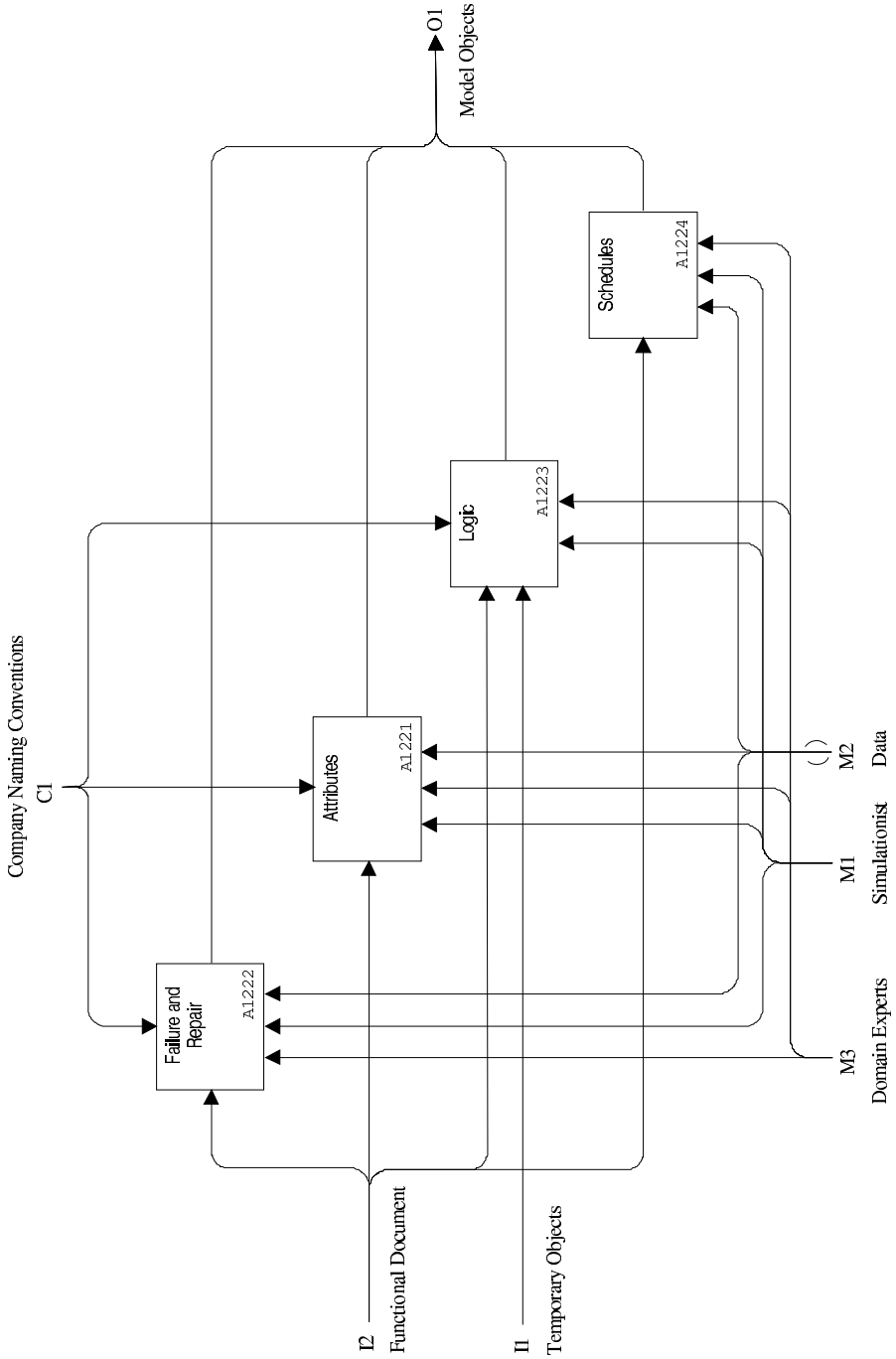


Figure A.9: *Node A122, Permanent Objects.*

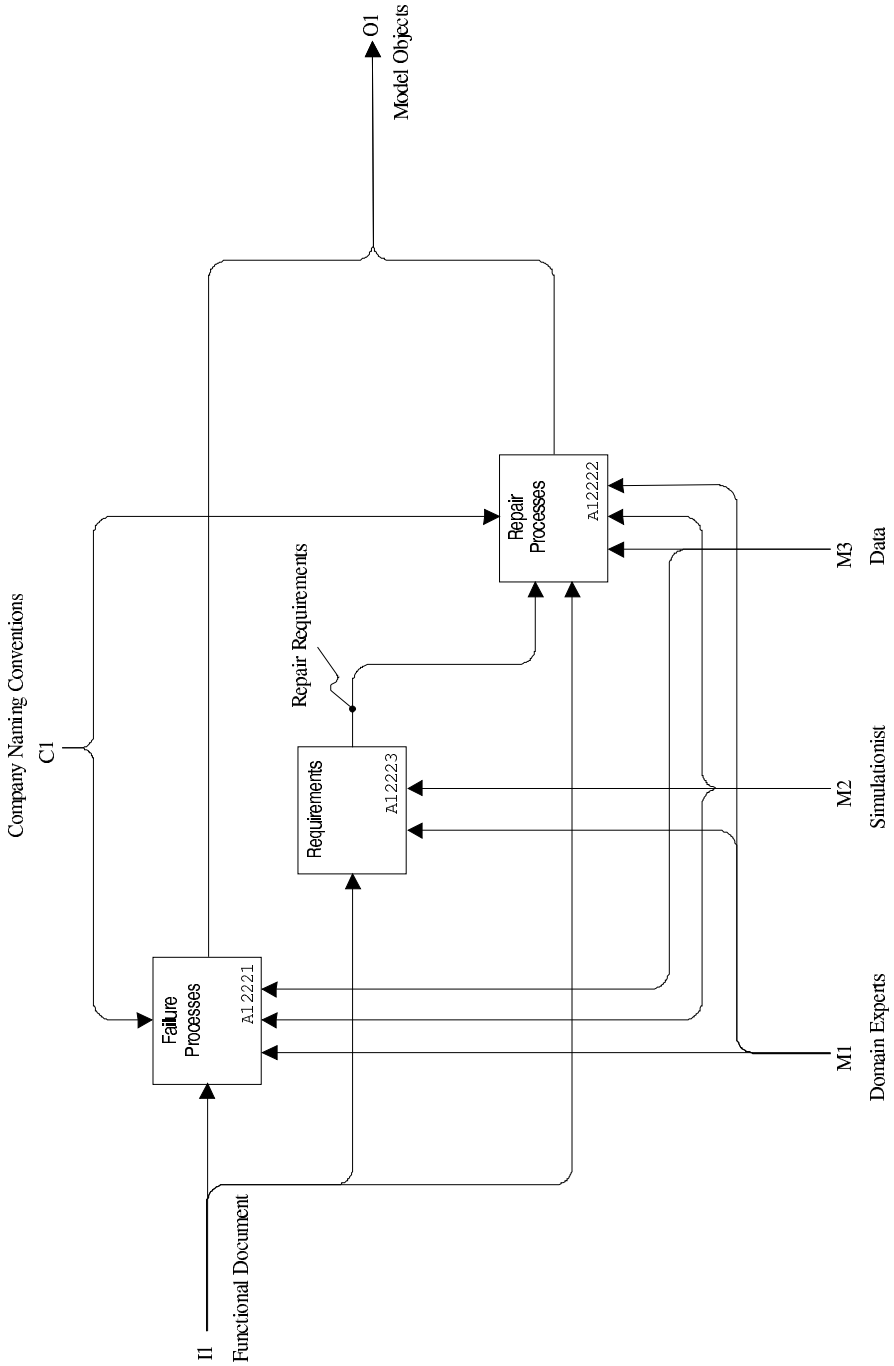


Figure A.10: *Node A1222, Failure and Repair.*

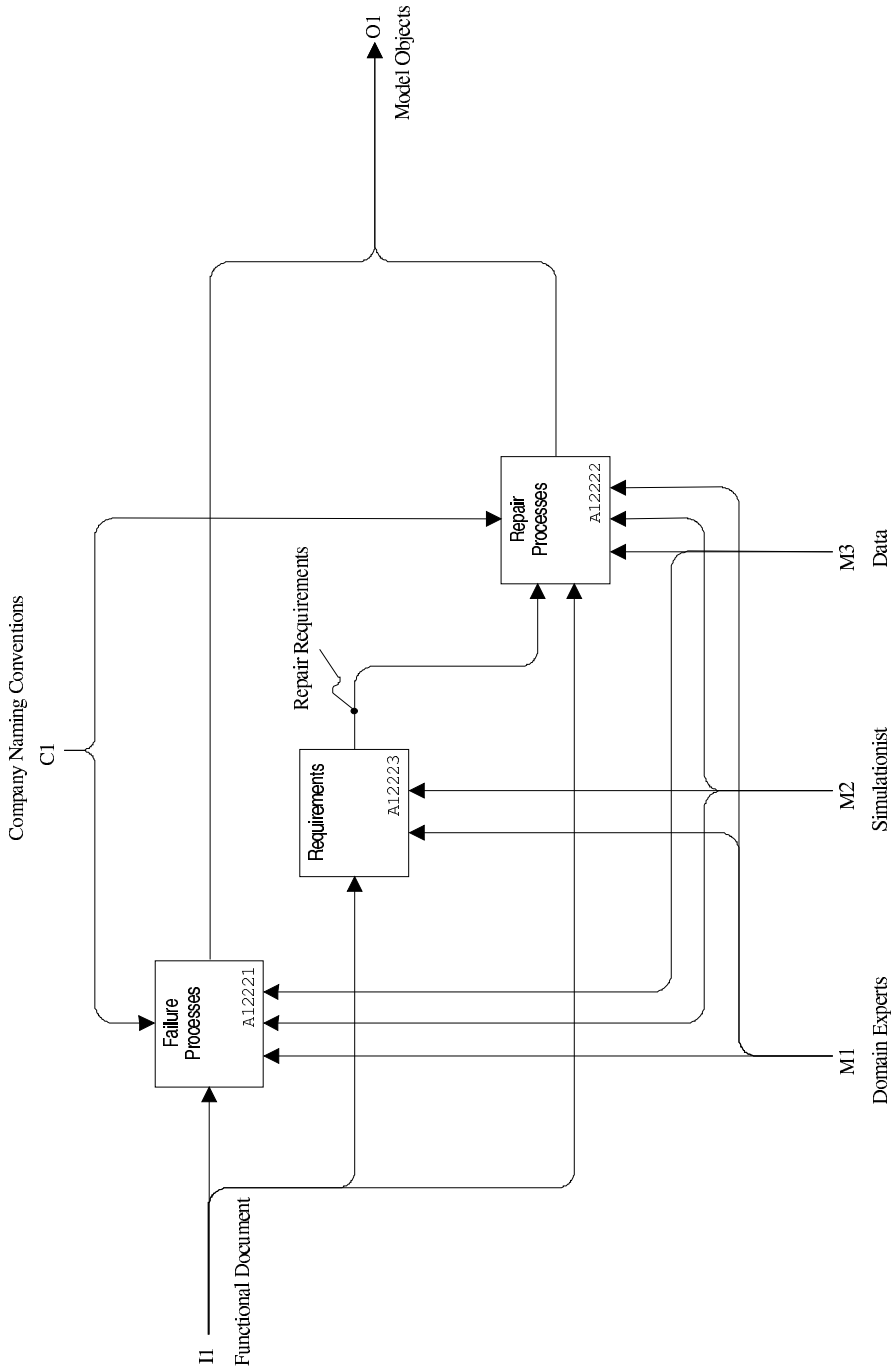


Figure A.11: *Node A123, Processes (Load/Unload/Setup/Cycle).*

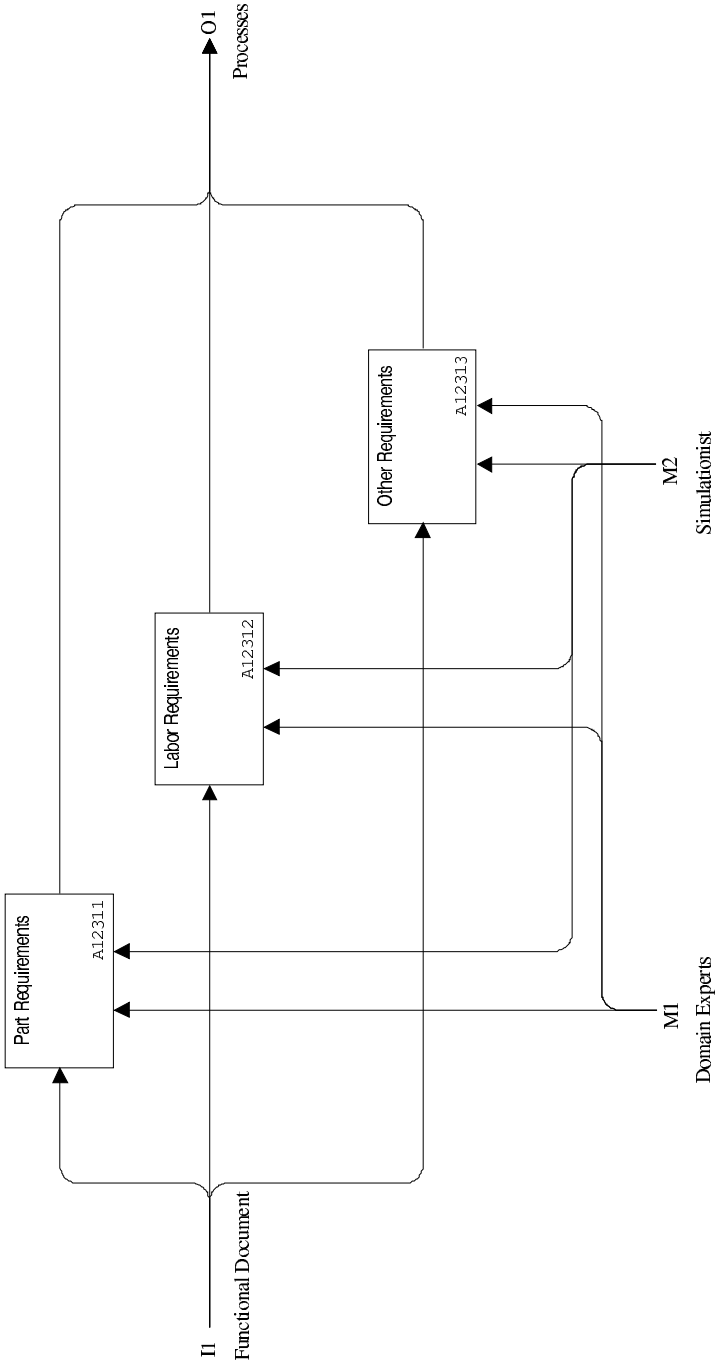


Figure A.12: *Node A1231, Requirements.*

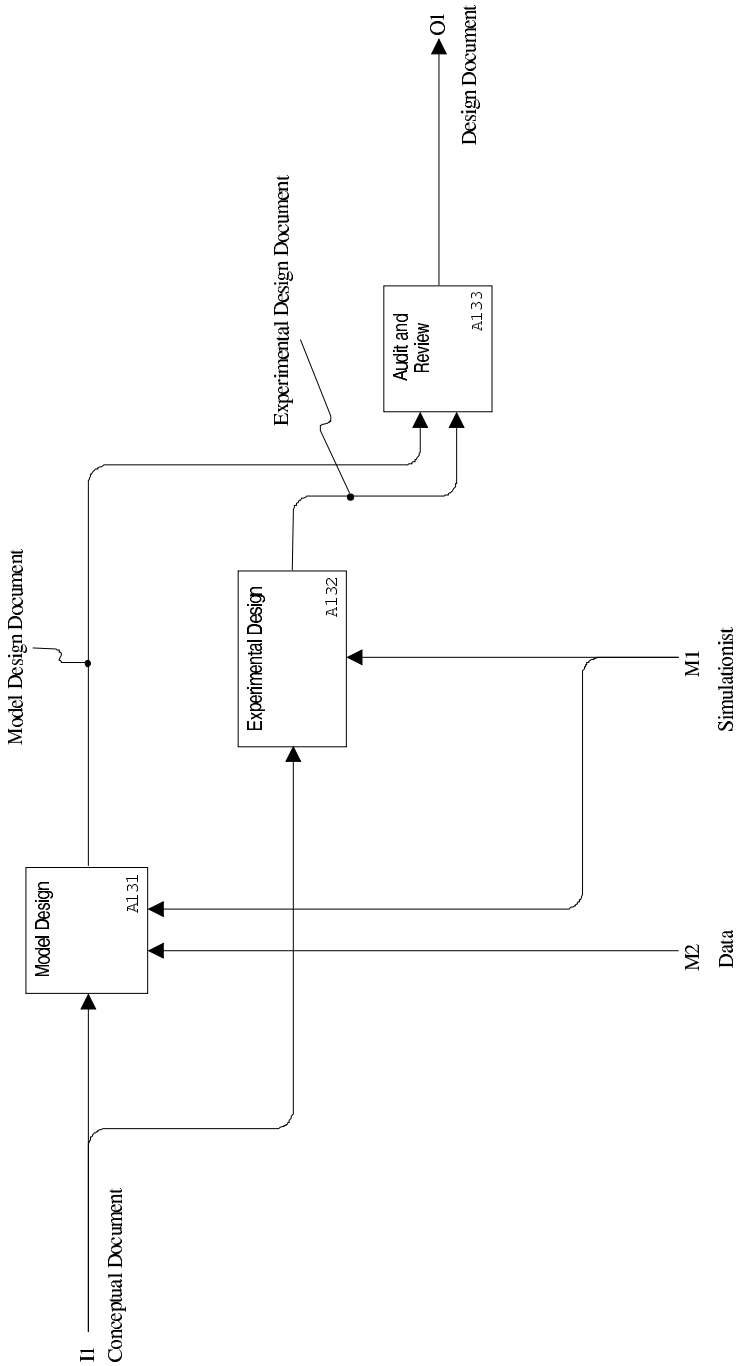


Figure A.13: *Node A13, Design Phase.*

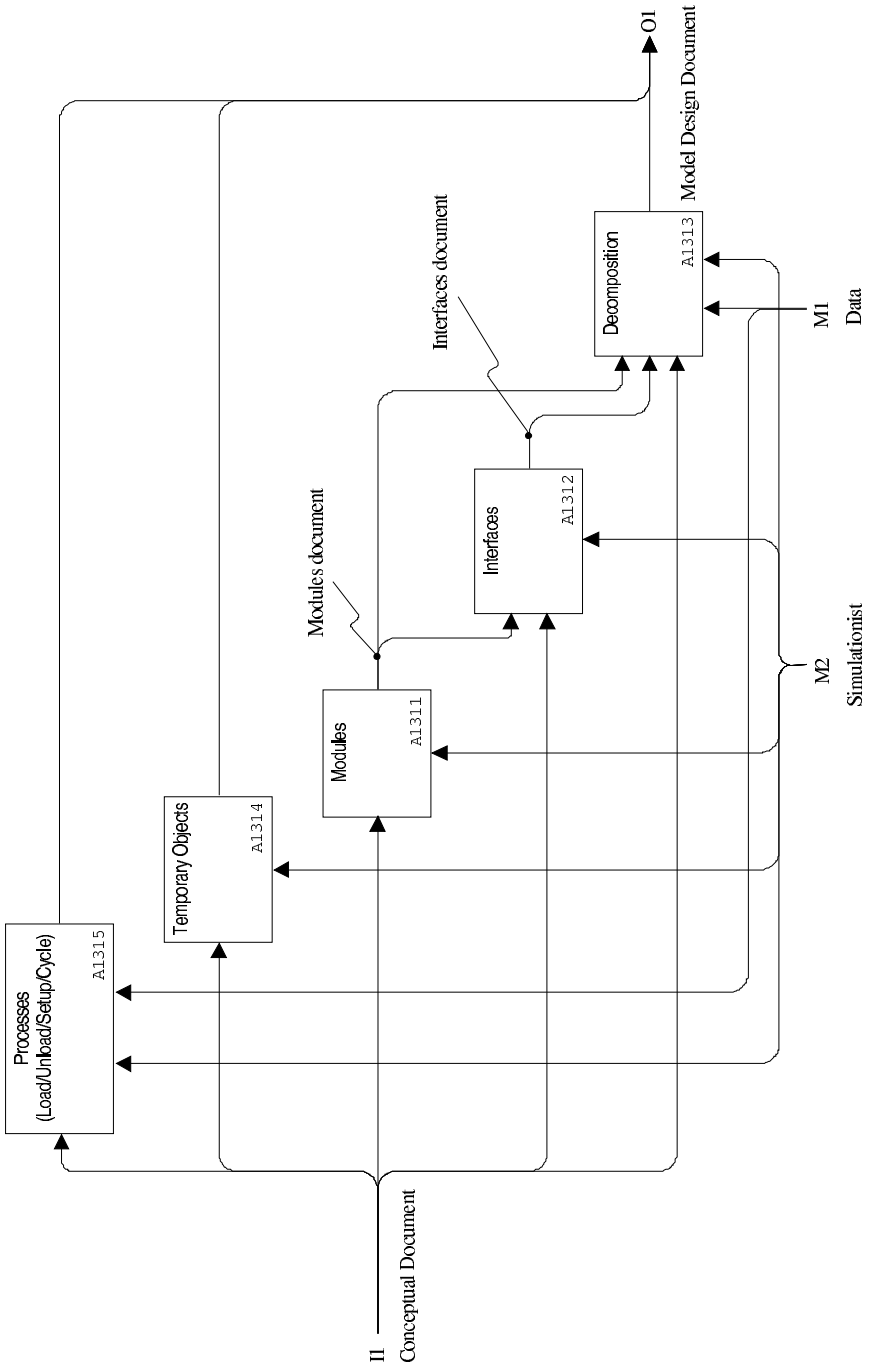


Figure A.14: *Node A131, Model Design.*

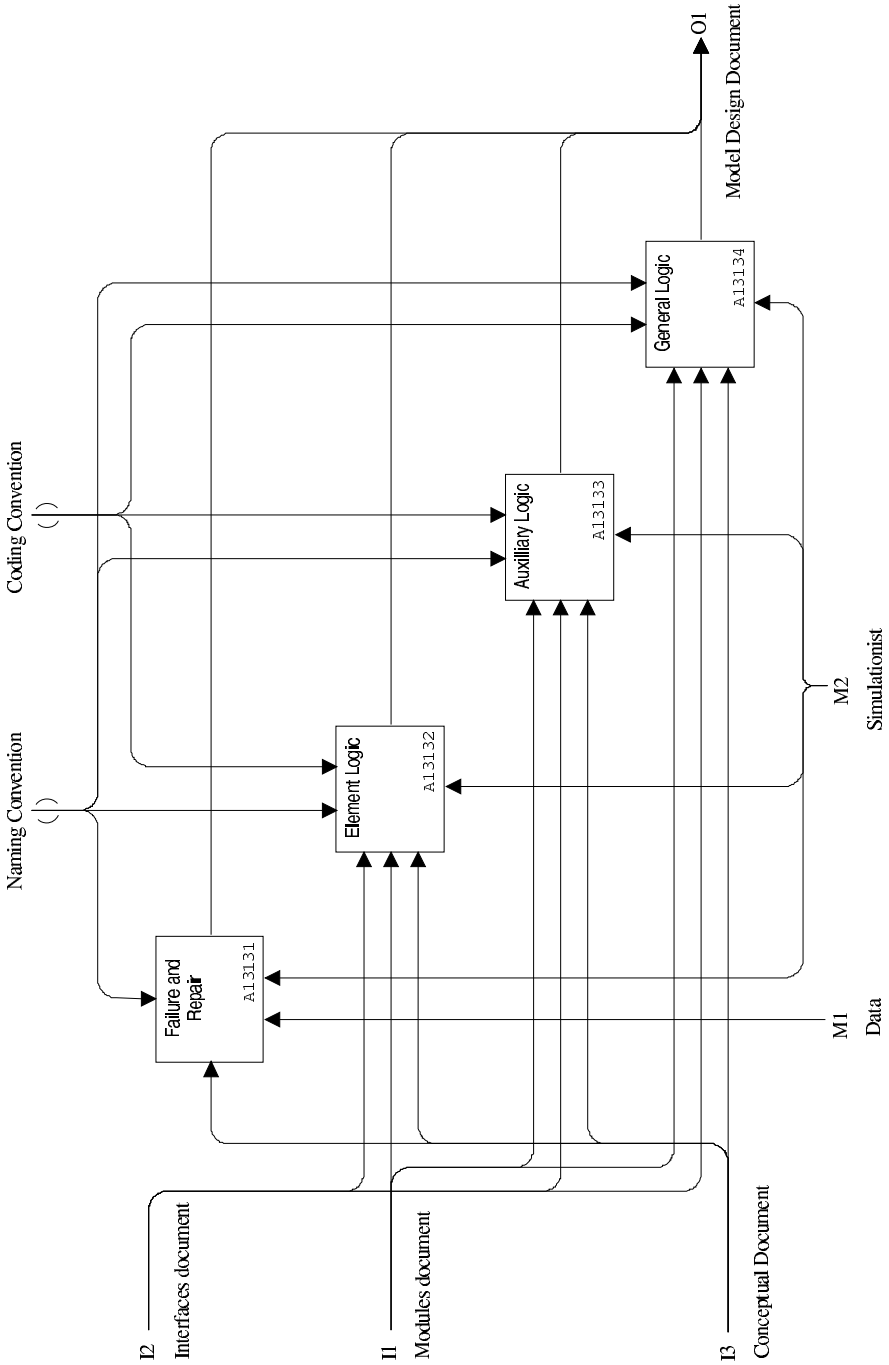


Figure A.15: *Node A1313, Decomposition.*

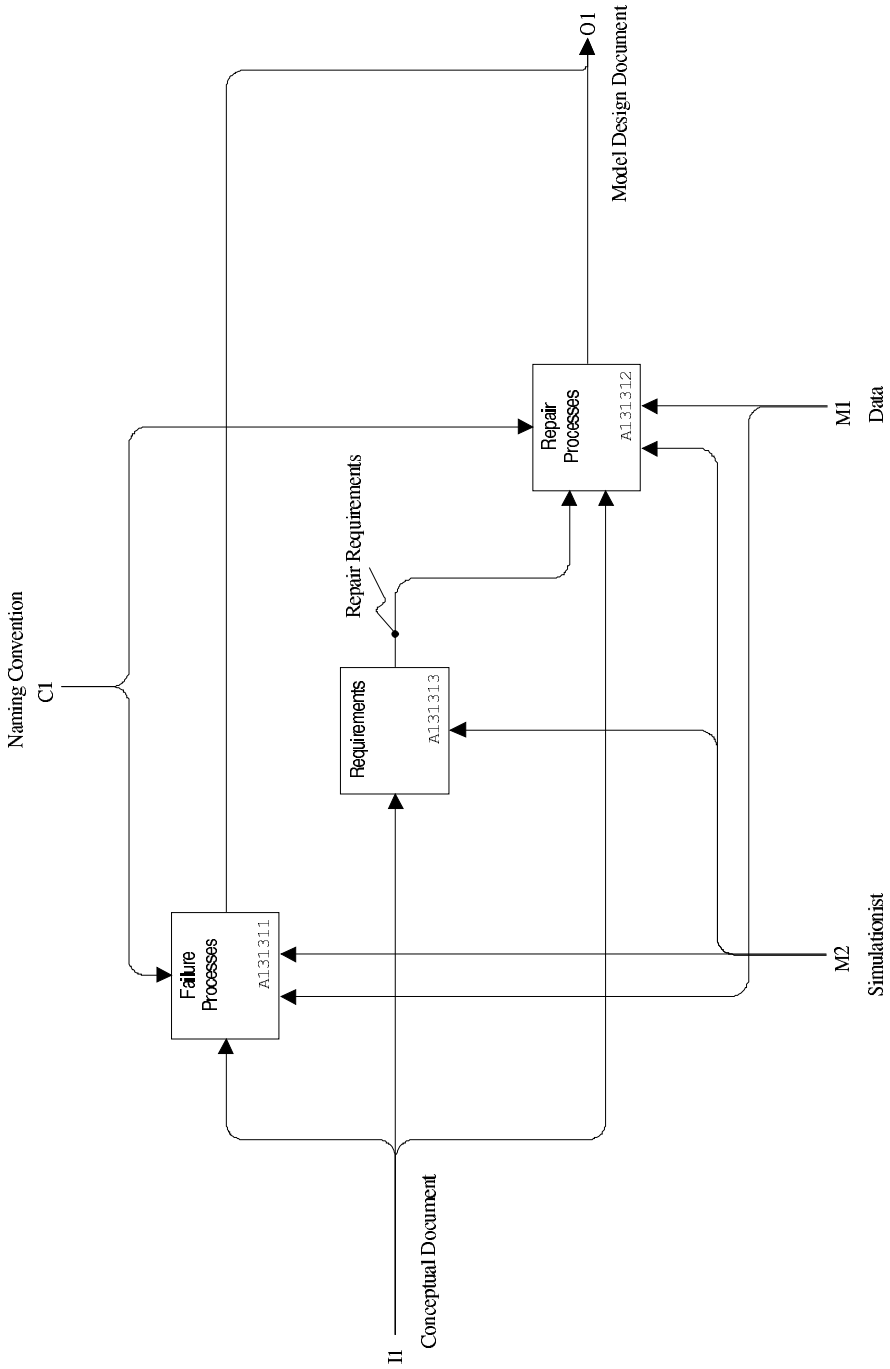


Figure A.16: *Node A13131, Failure and Repair.*

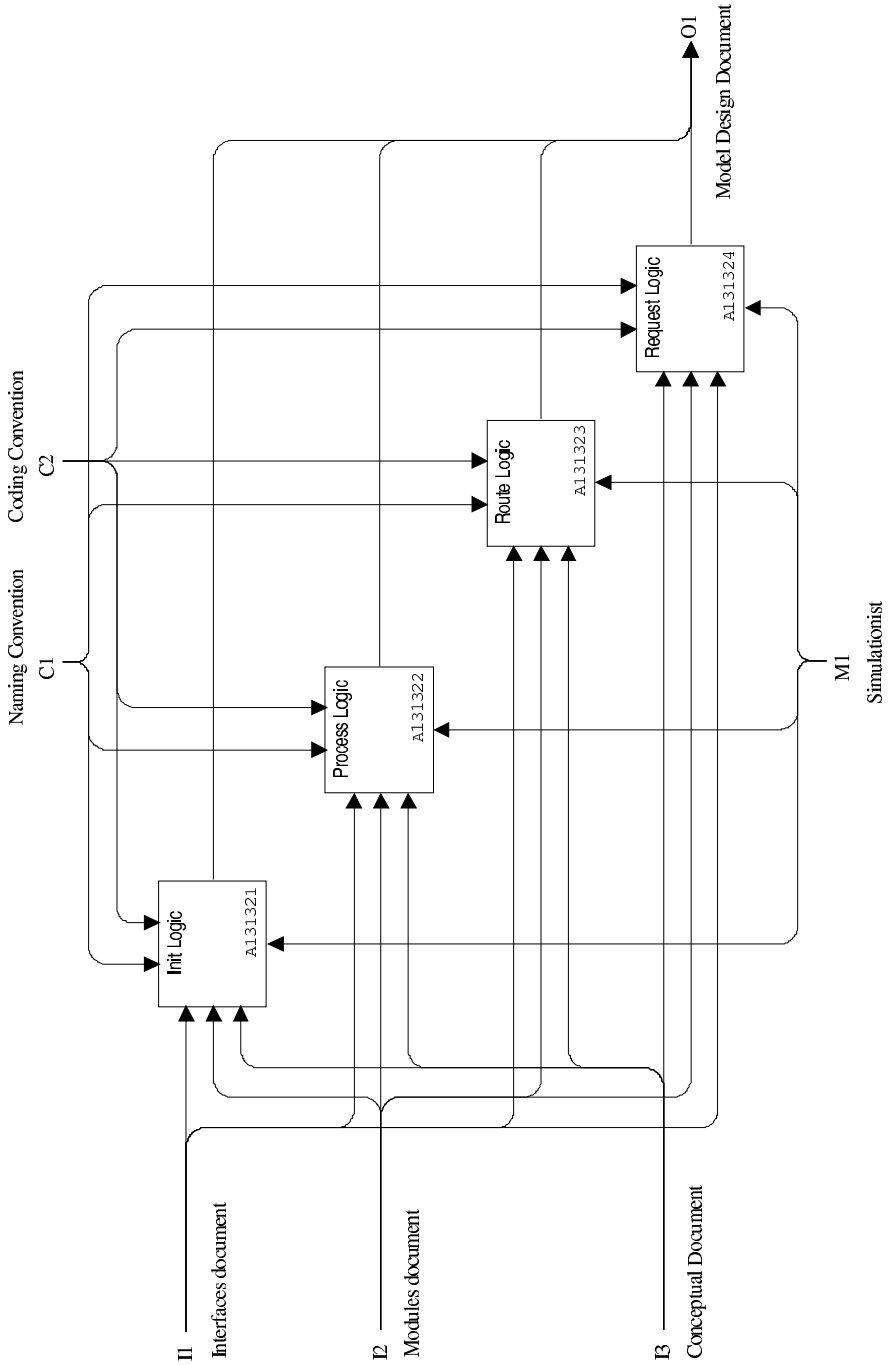


Figure A.17: *Node A13132, Element Logic.*

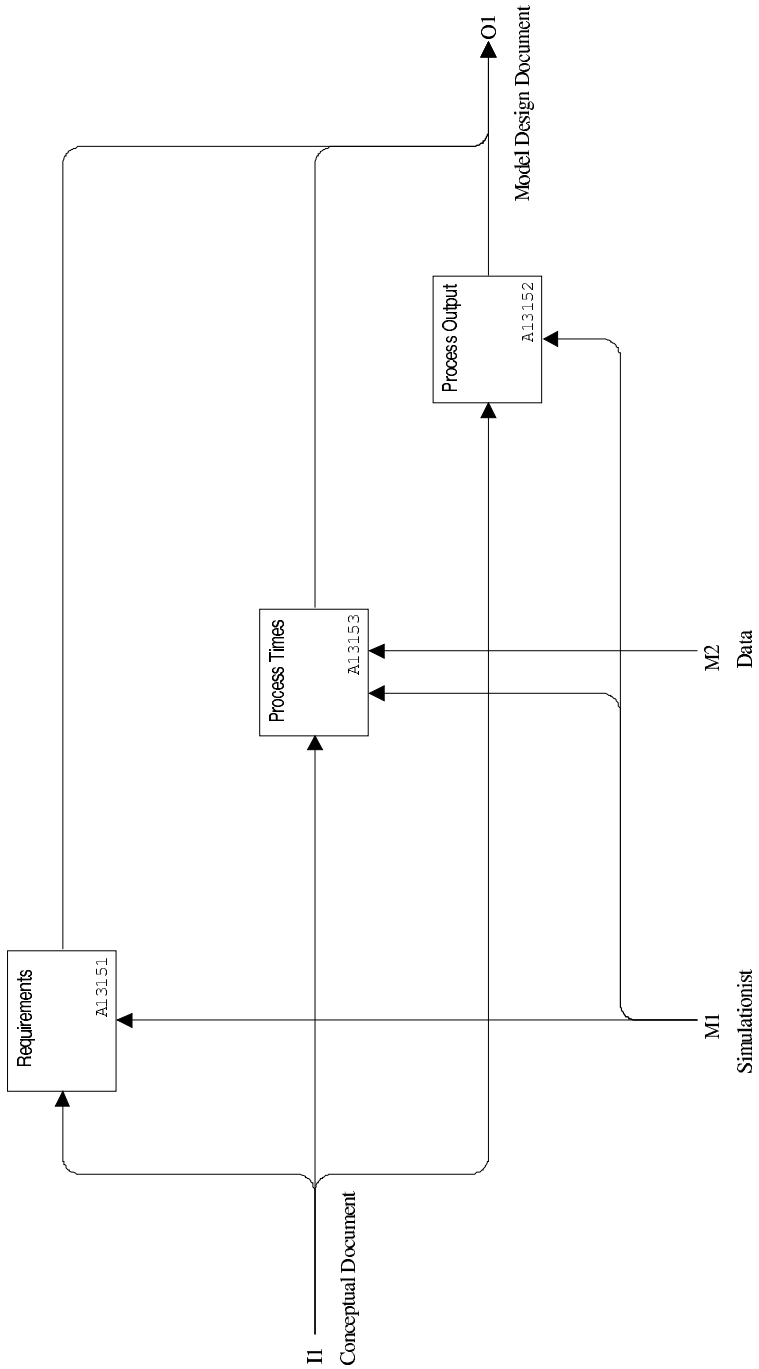


Figure A.18: *Node A1315, Processes (Load/Unload/Setup/Cycle).*

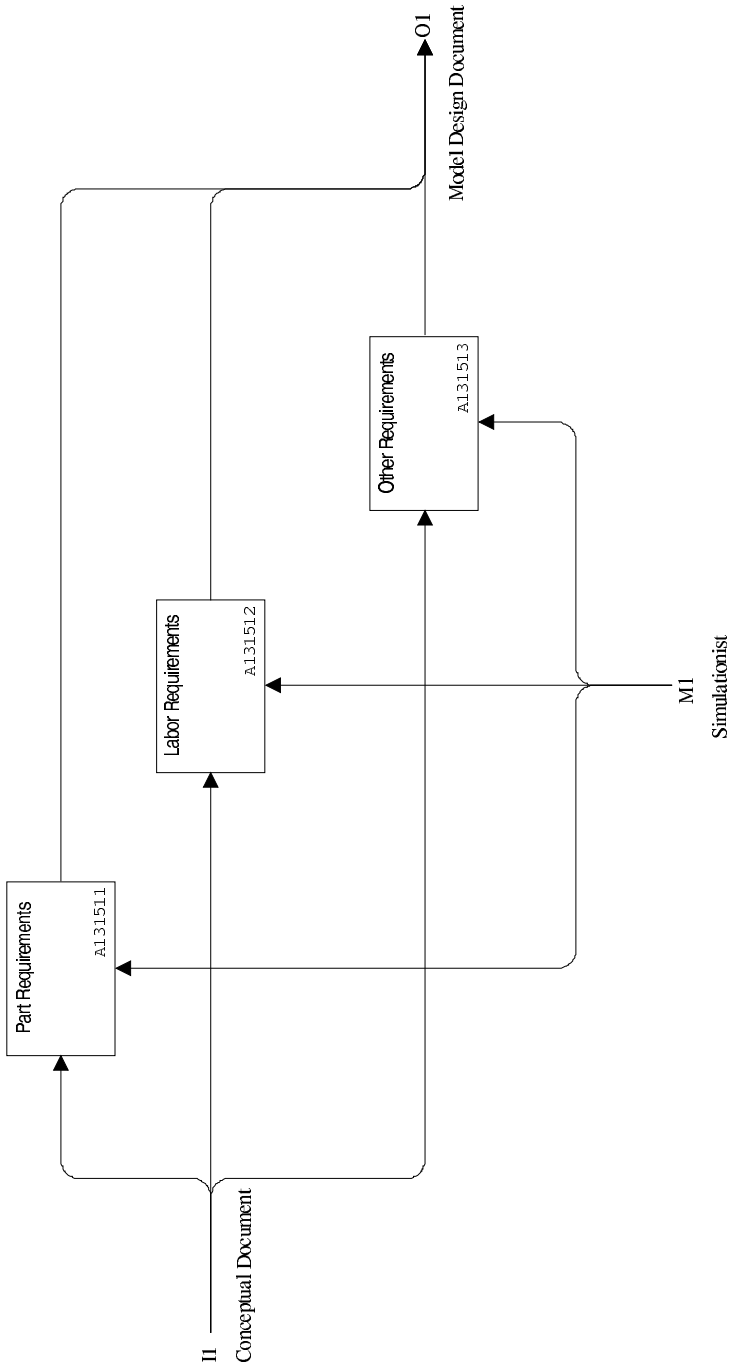


Figure A.19: *Node A13151, Requirements.*

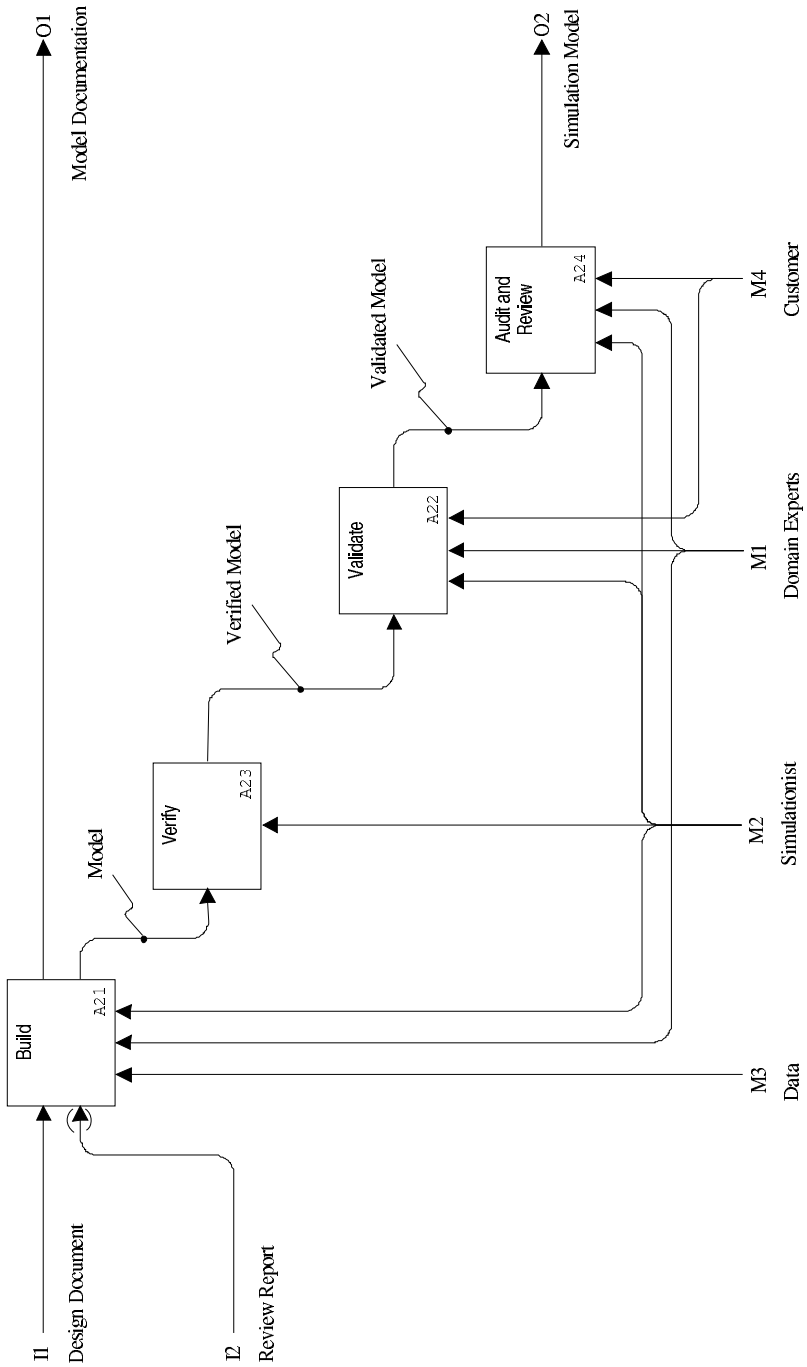


Figure A.20: *Node A2, Realization Phase.*

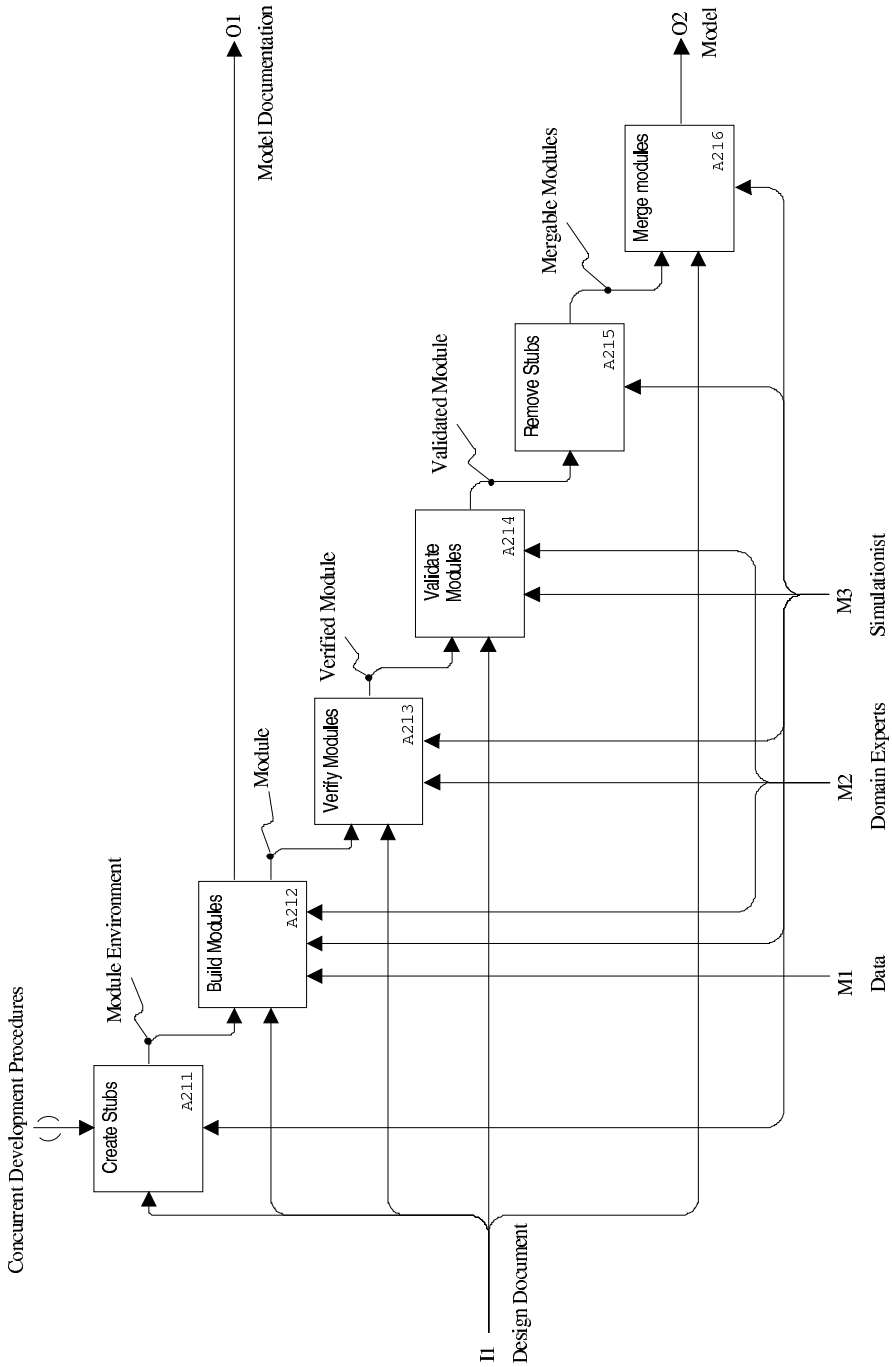


Figure A.21: *Node A21, Build.*

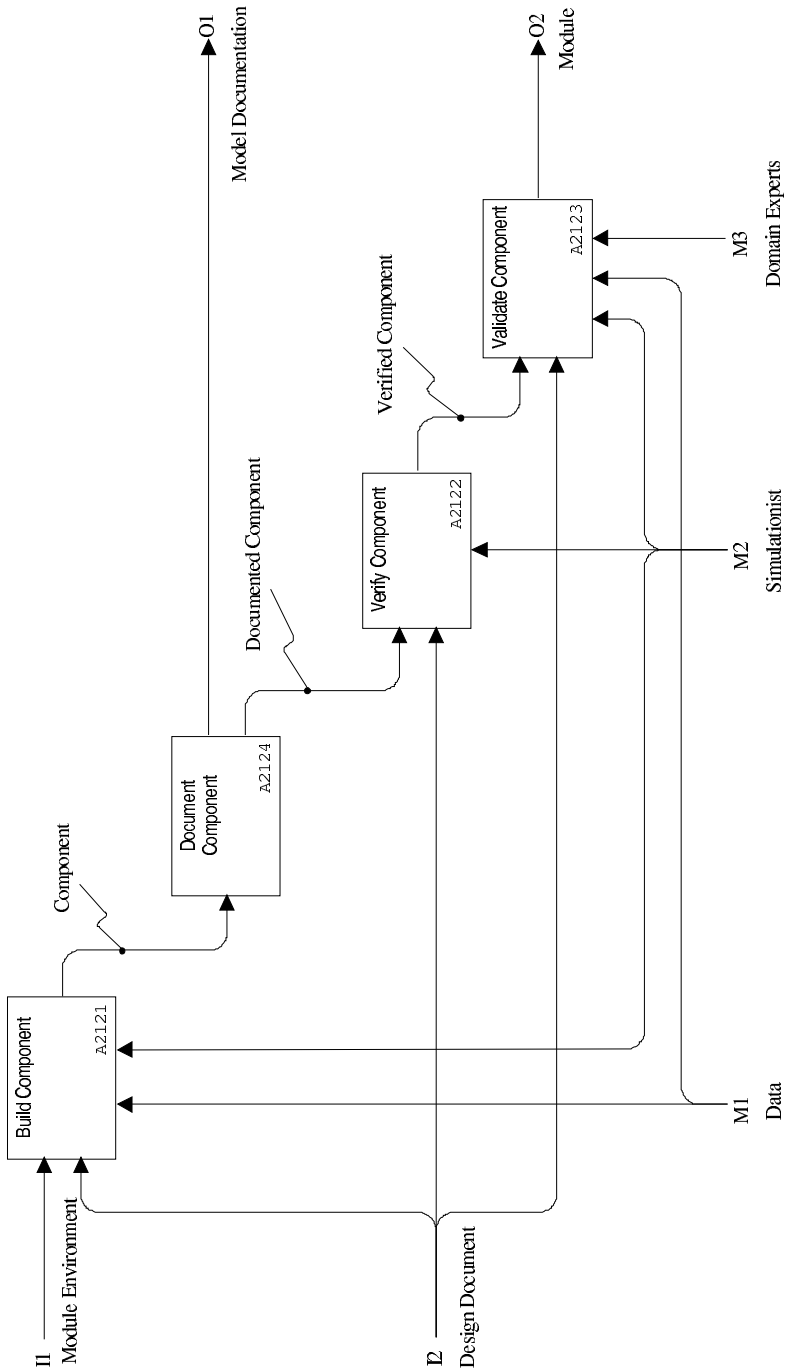


Figure A.22: *Node A212, Build Modules.*

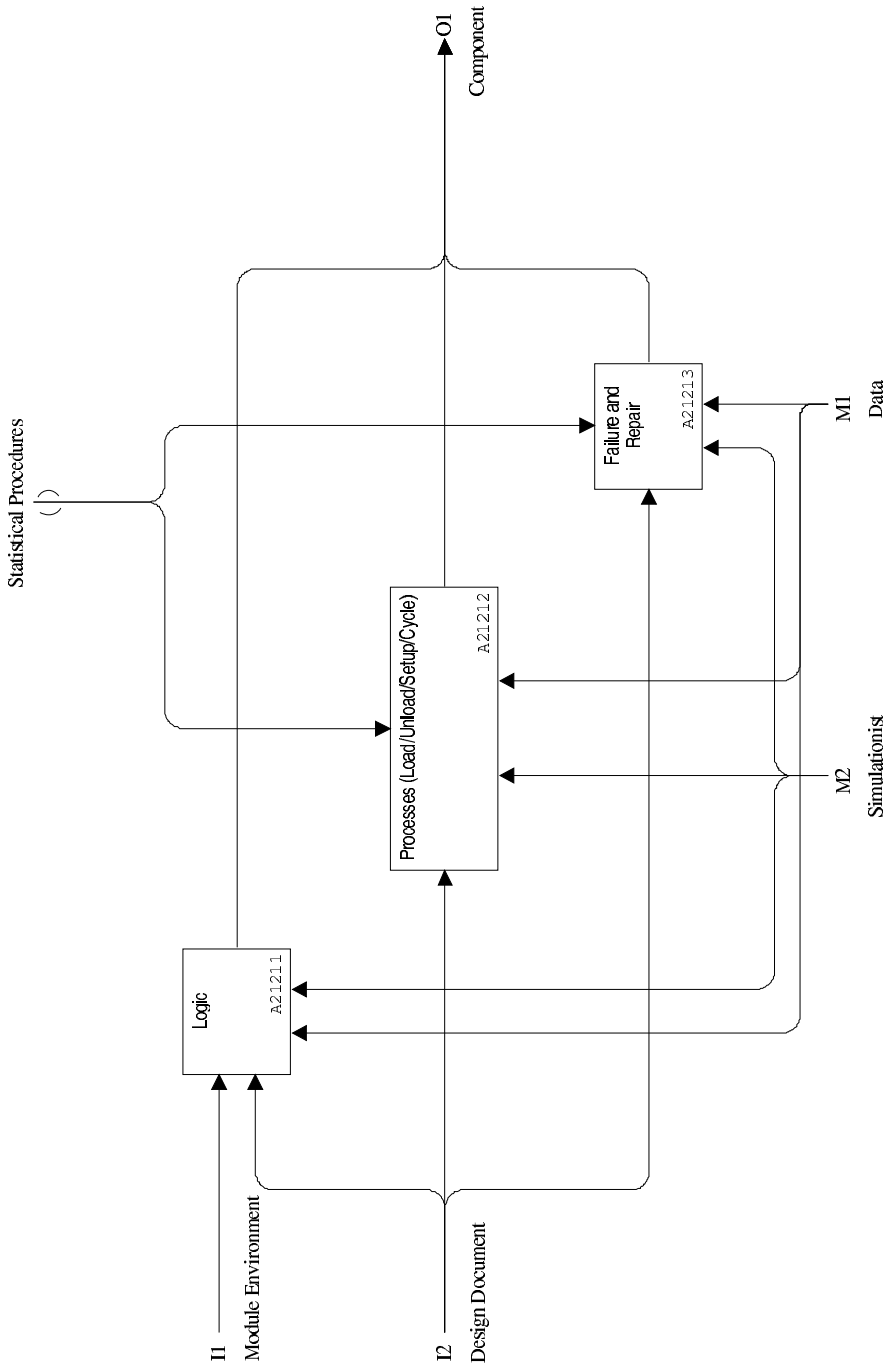


Figure A.23: *Node A2121, Build Component.*

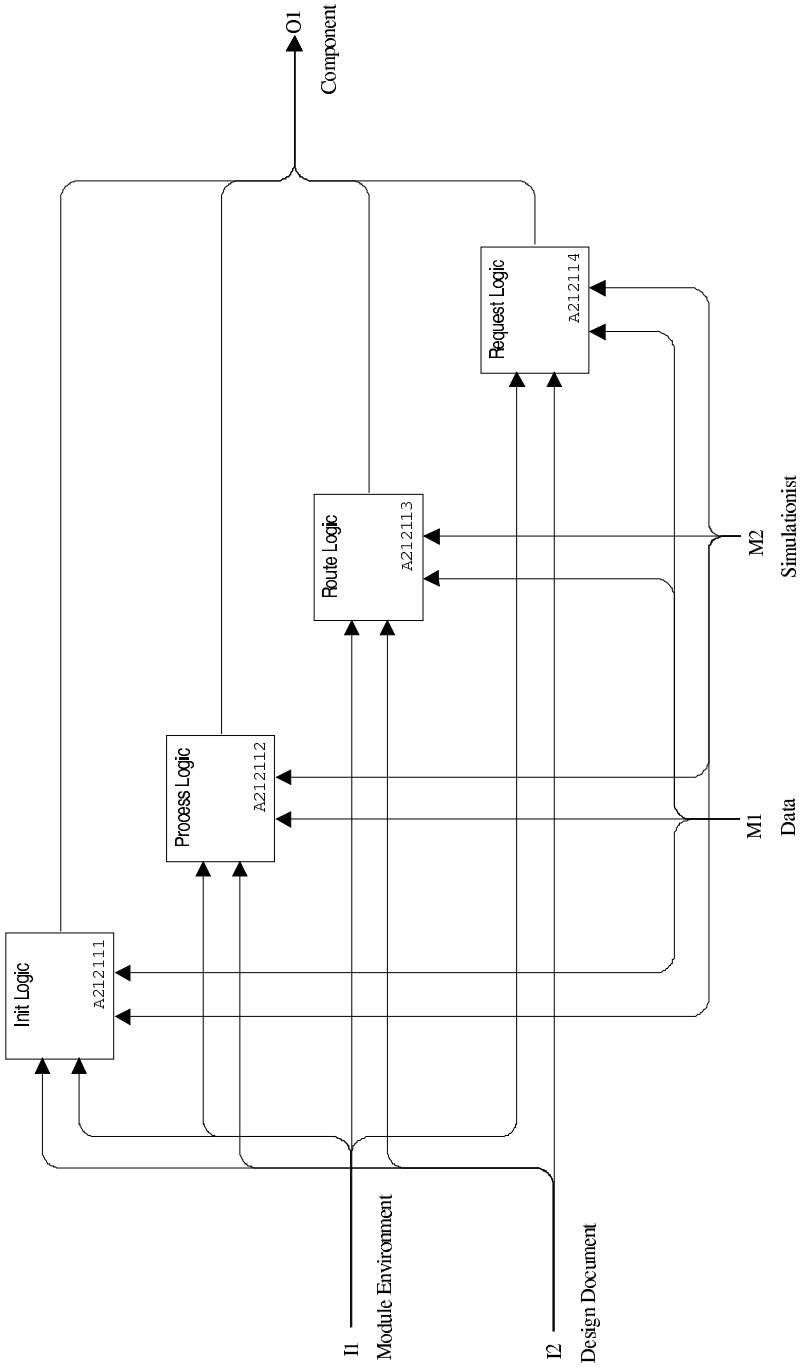


Figure A.24: *Node A21211, Logic.*

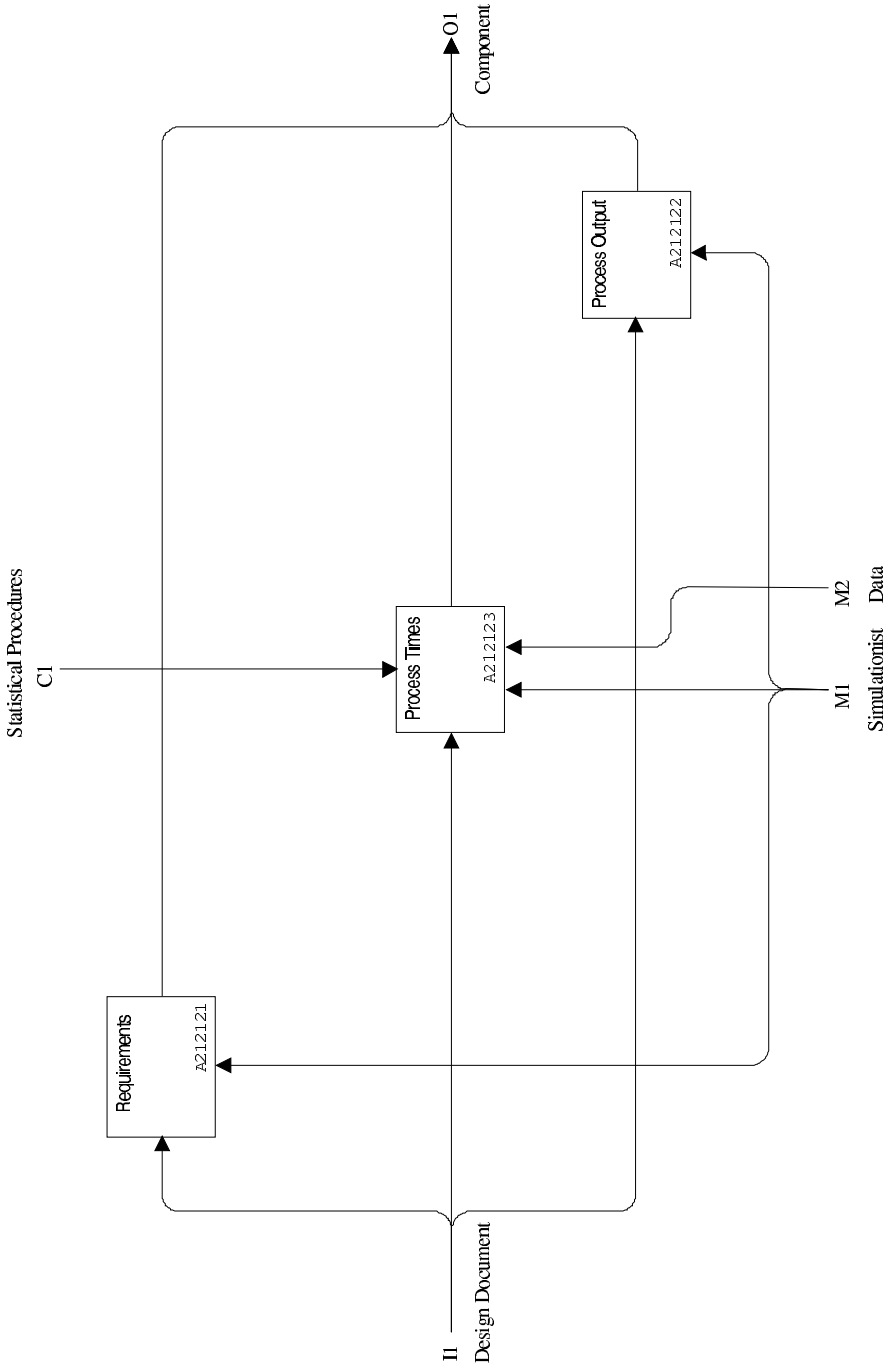


Figure A.25: *Node A21212, Processes (Load/Unload/Setup/Cycle).*

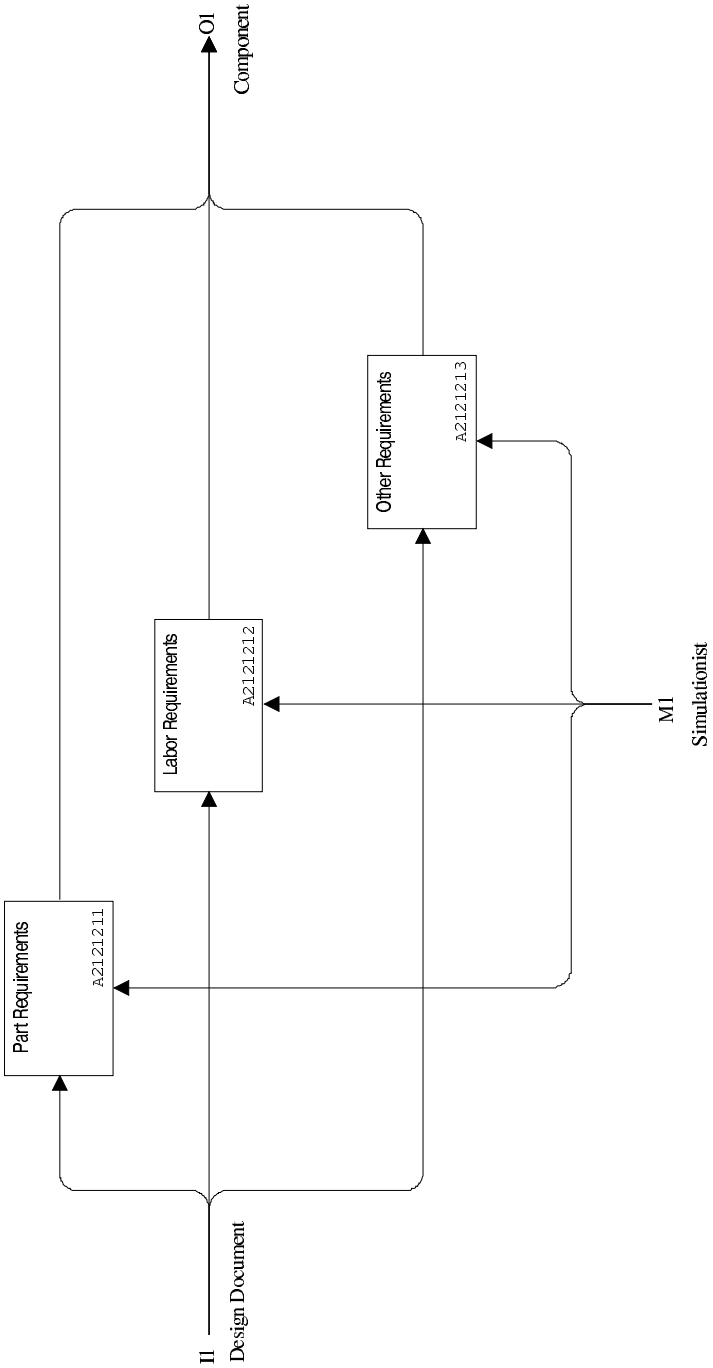


Figure A.26: *Node A212121, Requirements.*

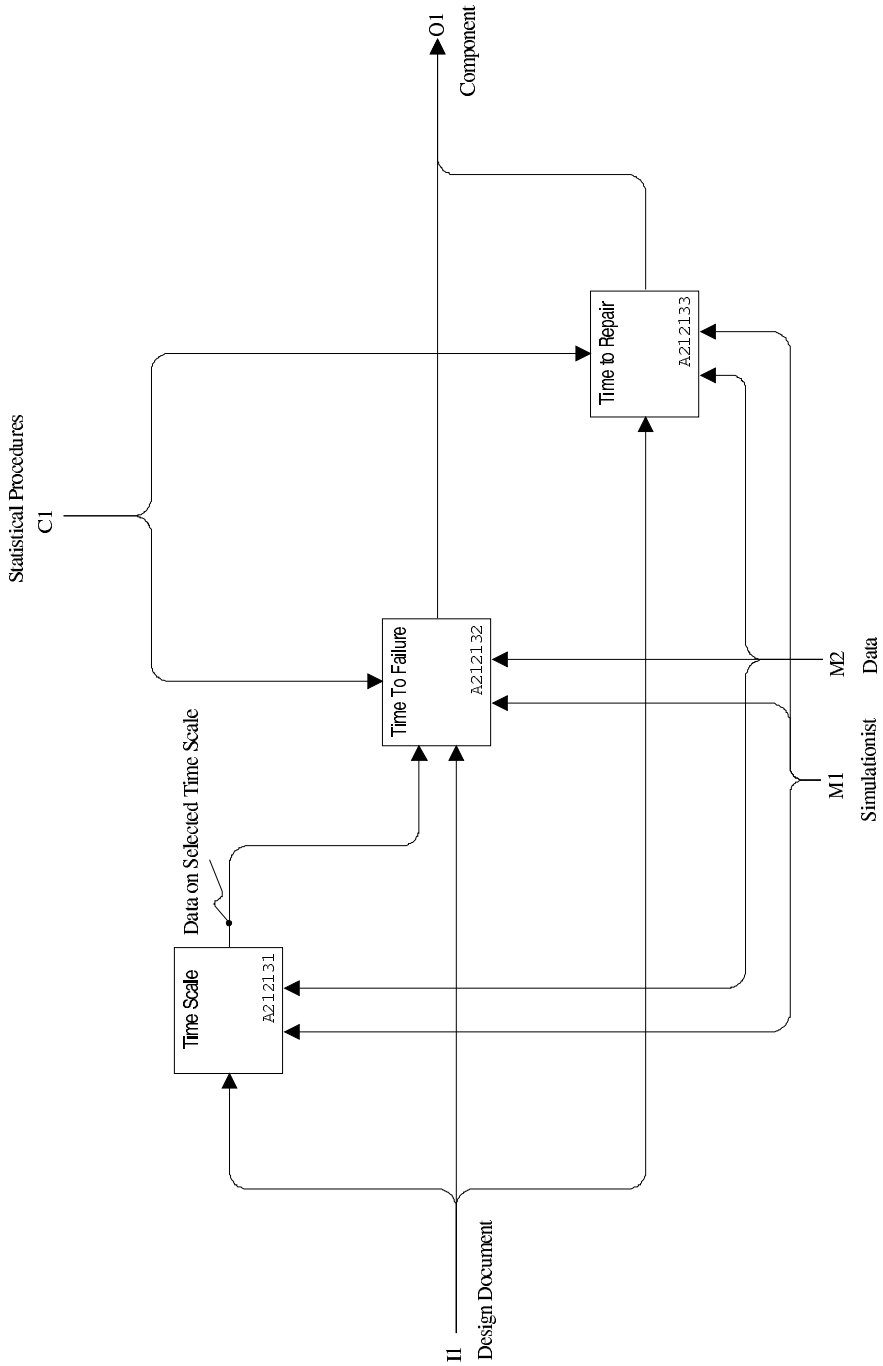


Figure A.27: *Node A21213, Failure and Repair.*

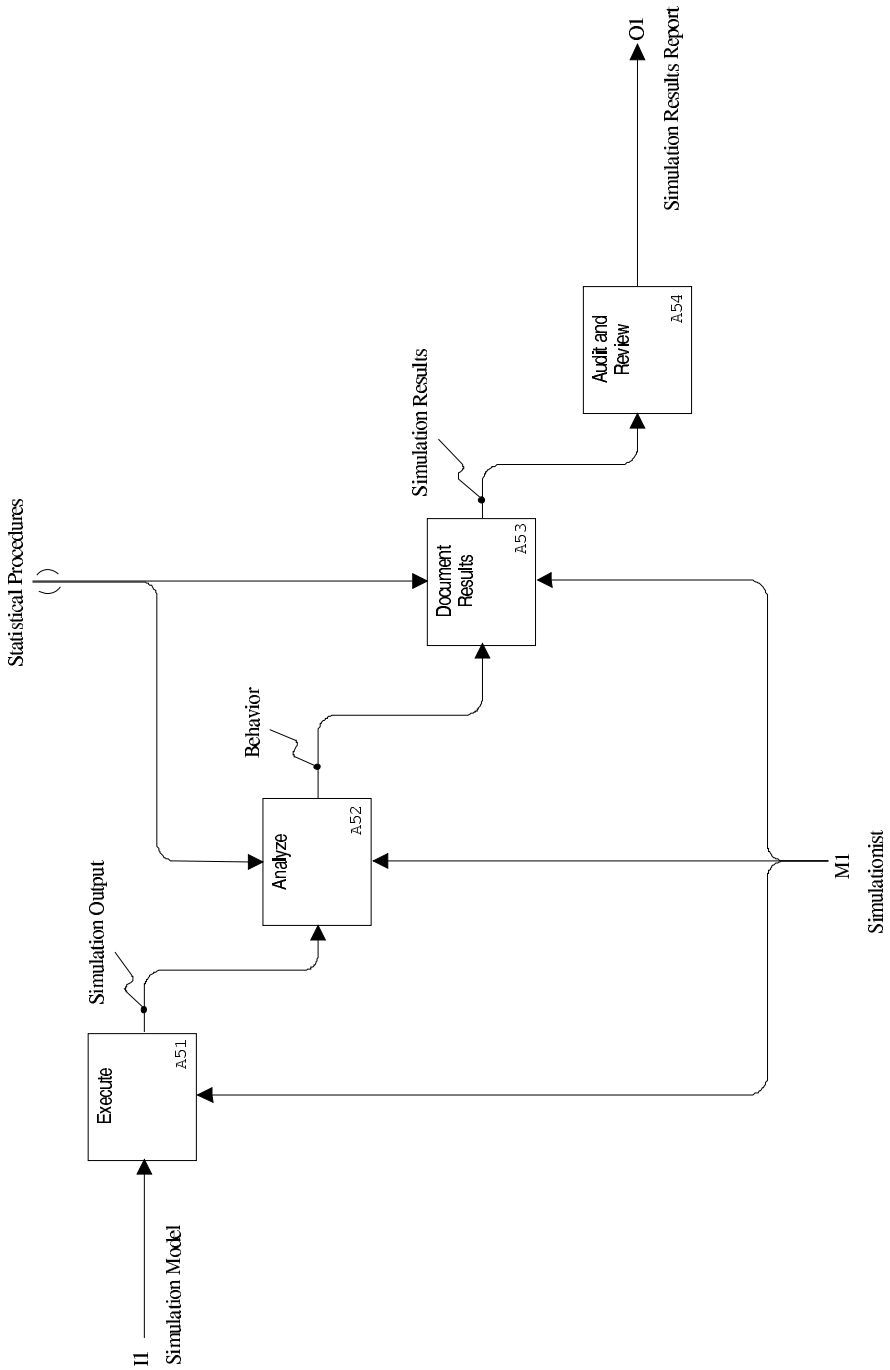


Figure A.28: *Node A5, Experimental Phase.*

Appendix B

Availability Related Definitions

The material presented here is mainly extracted from Lindgren & Rychlik (1997) and Lie, Hwang & Tillman (1977).

B.1 Basics

Definition B.1 *Lifetime*
The *lifetime* is the period of time from that an entity is started until it fails the first time.

Definition B.2 *Cumulative distribution function*
The cdf (cumulative distribution function) is defined as

$$F(t) = \int_0^t f(u)du = P(T \leq t) \tag{B.1}$$

where $f(u)$ is the pdf (probability density function).

Definition B.3 *Reliability function*
The probability that an entity will work at time t is

$$P(T > t) = 1 - F(t) = R(t) \tag{B.2}$$

where $R(t)$ is referred to as the *reliability function*.

Definition B.4 *Average lifetime*
The average lifetime is defined as

$$E(T) = \int_0^\infty u f(u)du. \tag{B.3}$$

Proposition B.1 *Mean time between failure and mean time to repair*
For every stochastic variable T with $P(T > t) = R(t)$

$$E(T) = \int_0^\infty R(t)dt \tag{B.4}$$

Definition B.5 *Failure rate*
If the lifetime T has the pdf $f(t)$, the cdf $F(t)$, and the reliability function $R(t)$, the *failure rate* is

$$z(t) = \frac{f(t)}{1 - F(t)} = \frac{f(t)}{R(t)} \tag{B.5}$$

Proposition B.2 *Reliability function*
If the lifetime distribution F has the failure rate $z(t)$, then

$$R(t) = \exp\left(-\int_0^t z(u)du\right) \tag{B.6}$$

Definition B.6 *Time between failure*
TBF is defined as the time between two downtimes excluding the repair time. TBF follows Equation 12.1

Definition B.7 *Time to repair*
The time an entity is in the repair state is referred to as TTR. This time excludes the WT. Planned maintenance is denoted TTR_p and corrective maintenance is denoted TTR_c

Definition B.8 *Downtime*
The downtime is the total time an entity is down and can be written as the sum of the WT (Ahlmann & Hagberg 190/91, Kiessling & Sandén 1981) and TTR

$$\text{DT} = \text{WT} + \text{TTR}.$$

When operators are modeled explicitly the WT should be excluded from the repair distribution. Downtime is thus dynamic and dependent on the number of operators in the system and the current system state. When operators are modeled implicitly the WT is included in the repair distribution. If possible, the WT and TTR should always be separated since all downtime distributions otherwise will be dependent of the number of operators which is undesirable.

B.2 Availability

Availability is defined in several ways in the literature (Ericsson 1997, Lie et al. 1977). Two classifications of the availability can be identified and are presented in the following sections.

B.2.1 Availability Based on the Time Interval

Depending on the time interval considered, availability is classified into the three categories:

- instantaneous,
- average uptime, and
- steady state availability.

In this context it is the steady state availability that is of interest.

Definition B.9 *Instantaneous availability*
Instantaneous availability, $A(t)$, is defined as the probability that the system is operational at any random time t. Compare this with the definition of reliability in definition B.3.

Definition B.10 *Average uptime availability*
Average uptime availability is the proportion of time in a specified interval $(0, T)$ that the system is available for use and is expressed as

$$A(T) = \frac{1}{T}\int_0^T A(T)dt \tag{B.7}$$

Definition B.11 *Steady state availability*
Steady state availability is defined as

$$A(\infty) = \lim_{T\to\infty} A(T) \tag{B.8}$$

B.2.2 Availability Based on Downtime Type

A classification can also be made considering the types of downtime:

- inherent,
- achieved, and
- operational availability.

In this category, the form used to describe system availability is that of an expected value function which assumes a steady state condition.

Definition B.12 *Inherent availability*
What most readers refers to as availability is here termed *inherent availability*, A_i, and is defined by

$$A_i = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}_c} \tag{B.9}$$

where MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair) follows from Equation B.4. It includes only corrective maintenance downtime and excludes ready time, preventive maintenance downtime, logistics time and waiting or administrative downtime.

Definition B.13 *Mean time between maintenance*

The MTBM (Mean Time Between Maintenance) is the mean interval of all maintenance requirements, i.e. both corrective and preventive. For example when preventive maintenance is scheduled at time T, it is expressed by

$$\text{MTBM} = \int_0^T R(s)ds \tag{B.10}$$

Definition B.14 *Mean maintenance time*

The *mean maintenance time*, M, is the downtime resulting from both corrective and preventive maintenance and is calculated by

$$\text{M} = \frac{\text{MTTR}_c f_c + \text{MTTR}_p f_p}{f_c + f_p} \tag{B.11}$$

where f_c and f_p respectively denote the number of corrective and preventive maintenance actions.

Definition B.15 *Achieved availability*

Achieved availability, A_a, includes corrective and preventive maintenance downtime and is a function of the frequency of maintenance and the mean maintenance time. It is expressed by

$$A_a = \frac{\text{MTBM}}{\text{MTBM} + \text{M}} \tag{B.12}$$

It excludes WT and thus both inherent and achieved availabilities are defined in an ideal support environment.

Definition B.16 *Operational availability*

Availability defined in an actual operational environment is termed the *operational availability*. Operational availability, A_o, includes WT and is expressed by

$$A_o = \frac{\text{MTBM} + \text{ready time}}{(\text{MTBM} + \text{ready time}) + \text{MDT}} \tag{B.13}$$

where

$$\text{ready time} = \text{operational cycle} - (\text{MTBM} + \text{MDT})$$
$$\text{MDT} = M + \text{WT}.$$

Operational availability appears to be a more realistic measure than the other two measures. Lie et al. argue that because delay time is determined by administrative and supply factors that can not accurately be anticipated, they are beyond the designer's control, and accordingly, can play little part in the maintainability of the design. However, within the framework presented here, this information is available and can thus be used in the development process.

Appendix C

An Example of the Incremental Development of Documents

Here a trivial example of the incremental generation of documents is presented for a few phases in the simulation study. The example below tries to replicate a fictive trivial simulation study. The information added in each phase is enclosed by XB ► and ◄ XB, where XB is an abbreviation for the baseline. The following abbreviations have been used:

FB Functional Baseline

CB Conceptual Baseline

DB Design Baseline

RB Realization Baseline

FB ►

Objectives

Build a model of a system with one machine and measure throughput.

◄ FB

CB ►

System Description

Temporary Objects

The IAT of `ThePart` is one minute following an exponential distribution.

Permanent Objects

The TheSource object creates ThePart and routes it to TheMachine. ◄ CB
DB ► The TheSource is a source class object. ◄ DB
CB ► TheMachine processes ten parts at a time. Each part takes one minute to process. Each batch of parts takes one minute to load and one minute to unload. There is no setup time. The parts are then sent to TheSink. ◄ CB
DB ► TheMachine is a machine class object and has the following processes attached:

Cycle process TheMachineCyc,

Load process TheMachineLoad, and

Unload process TheMachineUnload.

The process logic is defined by TheMachinePrc that is stored in TheMachine-Prc.scl file. ◄ DB
RB ►

TheMachinePrc

TheMachinePrc performs the process logic of the TheMachine.
SYNTAX:

```
procedure TheMachinePrc()
```

AUTHOR: Lars Randell (lrandell@robotics.lu.se)
CVS: Revision: 1.3

```
procedure TheMachinePrc()
var
  ii : integer
begin
  require part ThePart
  DoLoadProcess(TheMachineLoad)
  for ii = 1 to 10 do
    DoCycleProcess(TheMachineCyc)
  endfor
  DoUnloadProcess(TheMachineUnload)
  pass()
end
```

◄ RB
DB ► The TheSink is a sink class object. ◄ DB

List of Figures

1.1 Reformation of the product realization process 4
1.2 Implementation of concurrent engineering with 3-T loops . . . 6
1.3 Parallel development process 7

3.1 Typical steps in a simulation study 28
3.2 Concepts of the waterfall methodology 29

4.1 Different modeling views of a system 35
4.2 Data planning model for the model interface specification . . . 38
4.3 The proposed METK system 39
4.4 The life cycle of an simulated automated system 45

6.1 The new factory at Profilgruppen 60

7.1 An overview of the system studied at Volvo 64
7.2 The stop codes in the studied system 67
7.3 Variant tree of the simulation models 69

8.1 The central part of the simulation model at BT 74
8.2 The increasing number of communication channels 76

9.1 The new truck mast system simulation model at BT 80

10.1 The five levels of process maturity with key process areas . . . 87
10.2 Model accuracy effects on the investment risk 96
10.3 Manufacturing system development process with or without simulation 100
10.4 Resource allocation in the simulation study 100
10.5 Integrated discrete-event simulation study 100
10.6 Incremental development of a simulation model 101
10.7 Directory structure example 108

11.1 Example of documented source code 116
11.2 Example of formatted CodeDoc output 117

12.1 Usages of data in the manufacturing system life cycle 120
12.2 Time-scales 123
12.3 Time Between for a set S_k on a time-scale 124
12.4 Indirect losses in a safety zone 126
12.5 The `zone_repair` routine 128

13.1 Pseudo code for the `server_handshake` routine 133
13.2 Pseudo code for the `client_handshake` routine 133
13.3 The robot zones where handshaking was required 134
13.4 Pseudo code for a typical machine 135
13.5 The transfer stations, controller and robots in a synchronized line 135
13.6 Pseudo code for the `ctrl_handshake` routine used in the transfer stations' process logic 136
13.7 Pseudo code for the `buf_handshake` routine used in the transfer line controller 137
13.8 Pseudo code for the `transfer_elem_prc` routine used in the transfer stations 137

14.1 The database based input method based on a copy of the ERP database 141
14.2 An example of the SQL commands used to load the database . 143
14.3 The process logic pseudocode of the resource class 145
14.4 The route logic pseudocode of the resource class 145

A.1 Node A-0, Discrete-event simulation process 182
A.2 Node A0, Discrete-event simulation process 183
A.3 Node A1, Fundamentals 184
A.4 Node A11, Functional phase 185
A.5 Node A111, What 186
A.6 Node A112, How, Who, When 187
A.7 Node A12, Conceptual Phase 188
A.8 Node A121, Temporary Objects 189
A.9 Node A122, Permanent Objects 190
A.10 Node A1222, Failure and Repair 191
A.11 Node A123, Processes (Load/Unload/Setup/Cycle) 192
A.12 Node A1231, Requirements 193
A.13 Node A13, Design Phase 194
A.14 Node A131, Model Design 195
A.15 Node A1313, Decomposition 196
A.16 Node A13131, Failure and Repair 197
A.17 Node A13132, Element Logic 198
A.18 Node A1315, Processes (Load/Unload/Setup/Cycle) 199
A.19 Node A13151, Requirements 200
A.20 Node A2, Realization Phase 201

A.21 Node A21, Build 202
A.22 Node A212, Build Modules 203
A.23 Node A2121, Build Component 204
A.24 Node A21211, Logic 205
A.25 Node A21212, Processes (Load/Unload/Setup/Cycle) 206
A.26 Node A212121, Requirements 207
A.27 Node A21213, Failure and Repair 208
A.28 Node A5, Experimental Phase 209

List of Tables

3.1 Simulation usage survey in Sweden 23
3.2 Simulation usage survey in Germany 24
3.3 The different simulation types used by the current users in Germany 24
3.4 Time consumption in a simulation project 29
3.5 Time consumption in a simulation project 29

7.1 Selected fields of typical entries from the PANDA database . . . 66
7.2 Results from the statistical analyzes 70

10.1 Key process areas assigned to process categories 89
10.2 Key process areas mapped into the DES process phases 111

12.1 Data collection fields with examples for one entity 127

14.1 The database tables and fields in the proof-of-concept model . . 144

Acronyms

ABB BiW ..... ABB Body-in-White

ABC ..... Activity Based Costing

ABM ..... Activity Based Management

AD ..... Axiomatic Design

AGV ..... Automated Guided Vehicle

AP ..... Application Protocol

In order to define implementable standard information models for specific industry needs the STEP standard contains a construct called AP (Application Protocol). The AP defines the context and scope for how to interpret and use the data models defined in the Integrated Resources (Johansson 2001).

AQNM ..... Analytical Queuing Network Model

Simulations that provides quick estimates of steady state results regarding total system output and average resource utilization (Grewal et al. 1998).

ASCII ..... American Standard Code for Information Interchange

A character encoding.

BCL ..... Batch Control Language

Used with QUEST to run experiments, dynamically alter models etc.

BP ..... Business Process

BT ..... BT Products

CAD ..... Computer Aided Design

CAM ..... Computer Aided Manufacturing

CAME ..... Computer-Aided Manufacturing Engineering

See also CAMSE.

CAMSE ..... Computer-Aided Manufacturing System Engineering

McLean (McLean 1993) defines CAMSE as "the use of computerized tools in the application of scientific and engineering methods to the problem of the design and implementation of manufacturing systems." See also CAME.

CAPP ..... Computer Aided Process Planning

CCB ..... Change Control Board or Configuration Control Board

Dart (1990) and **?** refers to this as the *Change Control Board* while Bersoff et al. (1980) use the term *Configuration Control Board*.

cdf ..... cumulative distribution function

CE ..... Concurrent Engineering

CIM ..... Computer Integrated Manufacturing

CM ..... Configuration Management

CMM ..... Capability Maturity Model

CN ..... Computer Network

CNC ..... Computer Numerical Control

COTS ..... Commercial Off-The-Shelf

CRP ..... Capacity Requirements Planning

CVS ..... Concurrent Versions System

CVS is a configuration management system in the public domain. CVS allow concurrent development in heterogeneous environments, at different geographical sites.

DBMS ..... DataBase Management System

DELMIA ..... Digital Enterprise Lean Manufacturing Interactive Applications

DELMIA develops a number of simulation tools and is a part of Dassault Systemes that .

DES ..... Discrete-Event Simulation

DFMA ..... Design For Manufacturing and Assembly

ERP ..... Enterprise Resource Planning

EWP ..... Enercon Windtower Production
A producer of towers for wind energy turbines.

FEM ..... Finite Element Analysis

FMS ..... Flexible Manufacturing System

GOT ..... Graphical Operator Terminal

GUI ..... Graphical User Interface

HTML ..... HyperText Markup Language
HTML is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page.

IAT ..... Inter Arrival Time

IEEE ..... Institute of Electrical and Electronics Engineers

IGES ..... Initial Graphics Exchange Specification
IGES is an **ANSI!** graphics file format for three-dimensional wire frame models.

IMES ..... Initial Manufacturing Exchange Specifications

IRT ..... Inter Request Time

IS ..... Information System

ISO ..... The International Organization for Standardization

IT ..... Information Technology
IT applied to business organisations can be split into two major areas, IS (Information Systems) and CN (Computer Networks).

LTS ..... Lot Tracking System

MES ..... Manufacturing Execution System

METK ..... Manufacturing Engineering ToolKit
Results from the METK project at NIST is described in (Iuliano 1995, Iuliano & Jones 1996, Iuliano et al. 1997, Iuliano 1997, **?**).

ML ..... Maximum Likelihood

MRP ..... Manufacturing Resource Planning or Manufacturing Requirements Planning
See also MRP I and MRP II.

MRP I ..... Manufacturing Requirements Planning

MRP I enables a company to calculate how many materials of particular types are required, and at what times they are required (?). See also MRP II.

MRP II ..... Manufacturing Resource Planning

MRP II enables companies to examine the engineering and financial implications of future demand on the business, as well as examining the materials requirements implications (?). See also MRP I.

MTBF ..... Mean Time Between Failures

MTBM ..... Mean Time Between Maintenance

MTTR ..... Mean Time To Repair

NIST ..... National Institute of Standards and Technology

OLP ..... Off-Line Programming

PDF ..... Portable Document Format

PDF is a file format that has captured all the elements of a printed document as an electronic image that you can view, navigate, print, or forward to someone else.

pdf ..... probability density function

PDM ..... Product Data Management

Perl ..... Practical Extraction and Report Language

Perl is a language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. It's also a good language for many system management tasks.

PLC ..... Programmable Logic Controller

PS ..... PostScript

Postscript is a programming language that describes the appearance of a printed page.

QDBC ..... Quest DataBase Connection

QFD ..... Quality Function Deployment

According to Akao (1990), QFD "is a method for developing a design quality aimed at satisfying the consumer and then translating the consumer's demand into design targets and major quality assurance points to be used throughout the production phase."

QUEST ..... Queuing Event Simulation Tool

QUEST is a discrete-event simulation software from DELMIA with 3D animation capabilities.

RS ..... Robot Simulation

SCL ..... Simulation Control Language

The simulation programming language in QUEST.

SCM ..... Software Configuration Management

SDX ..... Simulation Data Exchange

A file format for exchanging simulation models (Moorthy 1999).

SE ..... Simultaneous Engineering

SFCS ..... Shop Floor Control System

SGI ..... Silicon Graphics

The company's computer systems, ranging from desktop workstations and servers to supercomputers, deliver advanced computing and 3D visualization capabilities to scientific, engineering and large enterprises. SGI also creates software for design, Internet, and entertainment applications.

SIMA ..... Systems Integration of Manufacturing Applications

SQL ..... Structured Query Language

A database query lanquage.

STEP ..... Standard for the Exchange of Product Model Data

The ISO 10303 STEP is a set of ISO standards which provide for the exchange of engineering product data. These standards can be grouped into infrastructure components and industry specific information models. STEP covers a wide range of application areas and each area has its own part of of the standard application protocols. STEP also provides a method for implementation in file exchange and database access (Johansson & Rosén 1999).

TAMCAM .... Texas A&M Computer Aided Manufacturing Laboratory

TBF ..... Time Between Failures

TCP/IP ..... Transmission Control Protocol/Internet Protocol

The main protocol of the Internet.

TEC ..... Teknikcentrum Kronoberg

TTR ..... Time To Repair

UWAB ..... Urshults Werkstads AB

VAH ..... Volvo Articulated Haulers

VCBC ..... Volvo Car Body Components

WBS ..... Whole Business Simulator

A concept where the entire enterprise is simulated (Love & Barton 1996).

WIP ..... Work In Progress

WSC ..... Winter Simulation Conference

WT ..... Waiting Time

WYSIWIS ..... What-You-See-Is-What-I-See

A metaphor for a synchronous editor that allows multiple users to access and edit shared material simultaneously.

Glossary

Aegis Aegis is a transaction-based software configuration management system.

AutoMod A discrete-event simulation package from AutoSimulations.

AutoSched A simulation package for fabs from AutoSimulations.

CodeDoc A program for extracting documentation from source code.

deductive A conclusion received from common principles is deductive (Patel & Tebelius 1987). See also inductive.

Delfoi Delfoi provides e-business solutions and distributes DELMIA products in Scandinavia.

DocStrip DocStrip is a macro supplied with LaTeX and is used to extract documentation and code from a source.

ecological validity The possibility of generalizing from one context to another (Patel & Tebelius 1987).

EXPRESS EXPRESS is an object-flavoured information modeling language that was developed as part of the STEP product data exchange standard. In 1994 the language was approved as an ISO International Standard, namely ISO 10303-11:1994.

IDEFØ IDEFØ is a method designed to model the decisions, actions, and activities of an organization or system. IDEFØ is useful in establishing the scope of an analysis, especially for a functional analysis. As a communication tool, IDEFØ enhances domain expert involvement and consensus decision-making through simplified graphical devices. As an analysis tool, IDEFØ assists the modeler in identifying what functions are performed, what is needed to perform those functions, what the current system does right, and what the current system does wrong. Thus, IDEFØ models are often created as one of the first tasks of a system development effort (**?**).

IDEF3 The IDEF3 process description capture method provides a mechanism for collecting and documenting processes. IDEF3 captures precedence and causality relations between situations and events in a form natural to domain experts by providing a structured method for expressing knowledge about how a system, process, or organization works (**?**).

inductive A conclusion based on individual cases that connect to a principle or a common law, is inductive (Patel & Tebelius 1987). See also deductive.

intranet Browsing software in conjunction with information servers and an underlying network infrastructure allows a business enterprise to disseminate and share all manner of business information efficiently, rapidly, and with a uniform user interface. As implemented within a single enterprise — regardless of whether the enterprise is one facility or multiple, geographically separated facilities — this combination of information and information dissemination mechanisms is known as an *intranet* (**?**).

IRIX IRIX is a Unix System V flavor running on Silicon Graphics workstations.

Make Make provides a simple mechanism for maintaining up-to-date versions of programs that result from many operations on a number of files (Feldman 1979).

Matlab Matlab from MathWorks integrates mathematical computing, visualization, and a language to provide a flexible environment for technical computing.

model A *model* is a simplified or idealized description of a system, situation, or process, often in mathematical terms, devised to facilitate calculations and predictions. See also system.

MySQL MySQL is a very fast, multi-threaded, multi-user, and robust SQL database server. MySQL is free software. It is licensed with the GNU General Public License.

PANDA PANDA is a computer system for sampling production data from PREVAS. Included in the system is also report generators.

population validity The possibility to generalize to a larger group (Patel & Tebelius 1987).

Scania The core of Scania's operations is the development, production and marketing of trucks for heavy transport work and buses and coaches for more than 30 passengers.

simulation The technique of imitating the behavior of some situation or system (economic, mechanical, etc.) by means of an analogous model, situation, or apparatus, either to gain information more conveniently or to train personnel.

simulator An apparatus for reproducing the behavior of some situation or system; esp. one that is fitted with the controls of an aircraft, motor vehicle, etc., and gives the illusion to an operator of behaving like the real thing.

system A *system* is defined as a group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose, see also model.

theory A scientific *theory* comprise usually something more than a hypotheses. A theory can be said to be a system of hypothesis (Patel & Tebelius 1987). See also **hypothesis!**.