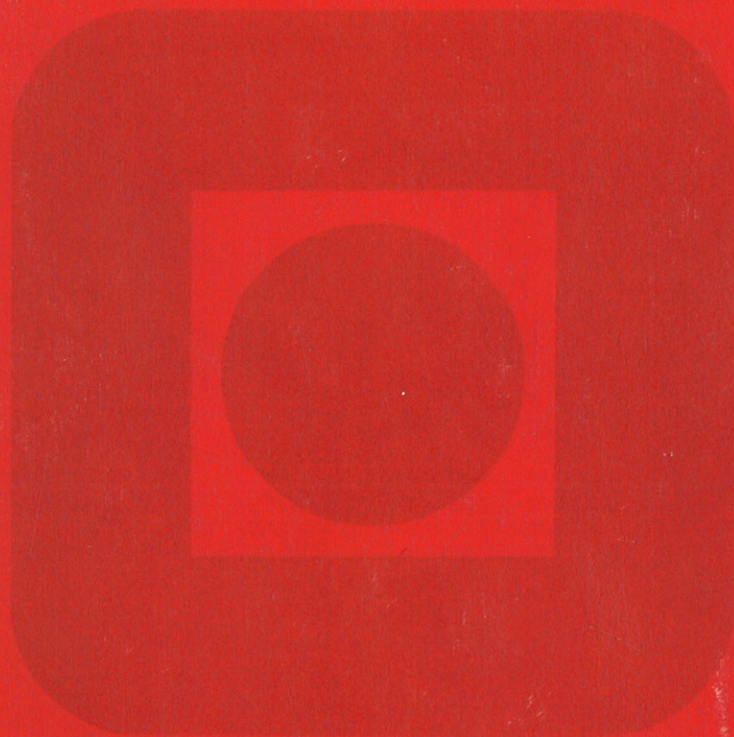


GUNNAR BRATAAS

PERFORMANCE ENGINEERING METHOD
FOR WORKFLOW SYSTEMS:
AN INTEGRATED VIEW OF HUMAN
AND COMPUTERISED WORK PROCESSES



NTNU

Norges teknisk-
naturvitenskapelige universitet
Trondheim

DOKTOR INGENIØRAVHANDLING 1996:62
INSTITUTT FOR DATATEKNIKK OG TELEMATIKK
TRONDHEIM

IDT-rapport 1996:5

651.5, 06:688, 502 3573.0

19566

Performance Engineering Method
for Workflow Systems:
An Integrated View of Human
and Computerised Work Processes

NORGE'S TECHNICAL UNIVERSITY

Gunnar Brataas

Information Systems Group
Faculty of Physics, Informatics and Mathematics
Norwegian University of Science and Technology
Gunnar.Brataas@idt.unit.no

This thesis is submitted in partial fulfilment
of the requirements for the academic degree of
Doktor ingeniør

4th July 1996

Abstract

A *method* for designing workflow systems which satisfy performance requirements is proposed in this thesis. Integration of human and computerised performance is particularly useful for workflow systems where human and computerised processes are intertwined. The proposed *framework* encompasses human and computerised resources.

Even though systematic performance engineering is not common practice in information system development, current *best practice* shows that performance engineering of software is feasible, e.g. the SPE method by Connie U. Smith. Contemporary approaches to performance engineering focus on *dynamic* models of resource contention, e.g. queueing networks and Petri nets. Two difficulties arise for large-scale information systems. The first difficulty is to estimate appropriate *parameters* which capture the properties of the software and the organisation. The second difficulty is to maintain an *overview* of a complex model, which is essential both to guide the choice of parameters and to ensure that the performance engineering process is an integral part of the wider system development process.

The proposed method is based on the *static* performance modelling method Structure and Performance (SP) developed by Peter H. Hughes. SP provides a suitable bridge between contemporary CASE tools and traditional dynamic approaches to performance evaluation, in particular because it addresses the problems of parameterisation and overview identified above.

The method is explored and illustrated with two case studies. The *Blood Bank Case Study* comprised performance engineering of a transaction-oriented information system, showing the practical feasibility of integrating the method with CASE tools. The *Gas Sales Telex Administration Case Study* for Statoil looked at performance engineering of a workflow system for telex handling, and consisted of performance modelling of human activity in interaction with a Lotus Notes computer platform. The latter case study demonstrated the feasibility of the framework.

Preface

This thesis is submitted to the Norwegian University of Science and Technology (NTNU) for the doctoral degree “doktor ingeniør”. The thesis advisors have been Professor Arne Sølvberg and Professor Peter Hughes. The research was carried out from March 1990 to March 1996 in the Information System group, Faculty of Physics, Informatics and Mathematics, Norwegian University of Science and Technology. The Norwegian University of Science and Technology (In Norwegian, Norges teknisk-naturvitenskapelige universitet, NTNU) was formed by the 1st of January 1996, as a merger between The Norwegian Institute of Technology (NTH) and other institutions.

I want to thank my advisors, Professor Arne Sølvberg and Professor Peter Hughes, for inviting me to this doctor’s degree study. Arne’s optimistic attitude and overview of thesis writing was of great help. Peter shared with me his deep knowledge of performance engineering.

Thanks are also given to Andreas Lothe Opdahl and Vidar Vetland for good cooperation in the IMSE project. Andreas inspired me with his conceptual overview and Vidar encouraged me to look at practical examples. Babak Amin Farshchian, Anne Helga Seltveit, Harald Rønneberg, and my colleagues in the Information Systems Group created a stimulating work environment. I also wish to thank the diploma students supervised by me; in particular Kenneth Finn Fløstrand, Håvard Dingstand Jørgensen and Alexander Kowalski. At the Regional Hospital in Trondheim, Ellen Berg, Knut Ekren, and Hilde Helgesen and their colleagues were helpful. Reidar Breivik, Ole Petter Drange, Grete Vika Lunde, Paul Nedrebø, Jan Rune Schøpp among others assisted me with the case study in Statoil. Eric Ole Barber gave valuable comments on some chapters.

Also thanks to my father Torbjørn Brataas, who commented on the English language, my mother Britt Sissel Foss Brataas, who has given encouragement, and my brother Arne Brataas, who will also finish his thesis soon. The members of the Acem-collective have formed a firm home base during this work. Thanks to Erlend Bleken and Asle Fostervold for chats, especially when prospects looked gloomy. Finally, thanks to Cecilie for getting me out of focus from time to time.

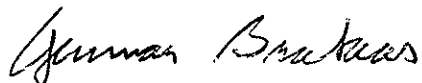
This work has been financed as follows: During 1990 – 91, I was engaged in the IMSE project. In 1992 – 93, I was granted a scholarship from The Norwegian Institute of Technology. My research in 1994 was supported by the BEST project, which was funded by The Research Council of Norway. Finally, I have worked as researcher in the Information System Group at NTNU during the last year of my thesis completion.

During the working period, the author has collaborated with a number of people. To make sure the thesis cites the source for anything that does not originate with the author, I have adopted the following conventions:

- Work done by others: the source of the contribution is cited.
- Work done in collaboration with others where it is difficult to distinguish who has contributed: it will be explicitly stated in the text that the work reported is done together with other people.
- Work done by M. Sc. students supervised by the author: the particular report is cited.
- Everything else is the author's own contribution where nobody else except the thesis advisors have been involved.

An outline of the thesis is given in Table 0.1.

Trondheim, 4th July 1996



Gunnar Brataas

Outline of the Thesis

Basic framework of this thesis is introduced Chapter 2. The framework is motivated by the problem with the interaction between the organisation and the computer system in the Blood Bank Case Study.

State of the art in the fields of:

- Classical performance evaluation (Chapter 3).
- The Structure and Performance (SP) method (Chapter 4).
- Performance engineering (Chapter 5).
- Organisational change and workflow technology (Chapter 8).

Method for:

- Performance engineering of information systems (Chapter 6).
- Performance engineering of workflow systems (Chapter 9).

Case studies for:

- Transaction-oriented information system: A more detailed analysis of the Blood Bank Case Study (Chapter 7).
- Workflow systems: The Gas Sales Telex Administration Case Study. This case study is divided in two parts:
 - Organisation: Performance modelling of the manual telex handling in the gas sales telex administration organisation, in Chapter 10.
 - Computerised workflow system: Chapter 11 describes elements of a model of the Lotus Notes workflow platform software in the Gas Sales Telex Administration Case Study. This chapter also describes the integration of the Lotus Notes platform with the manual processing as described in Chapter 10.

Conclusions and Further Work in Chapter 12.

Table 0.1: A brief outline of the thesis.

Contents

1	Introduction	1
1.1	Research Context	1
1.2	Investigating the Problem	2
1.3	Research Objectives	3
1.4	Claimed Contributions	4
2	Basic Framework	5
2.1	A First Case Study — The Blood Bank	5
2.2	Basic Framework	9
2.3	Applying the Basic Framework on the Transfusion Process	12
2.4	Performance Measures in Different Domains	13
2.5	Chapter Summary	14
3	Performance Modelling	15
3.1	Workload, System and Performance	15
3.2	Contention Modelling	19
3.3	Performance Modelling Cycle	23
3.4	Chapter Summary	23
4	Structure and Performance (SP)	25
4.1	Hierarchies of Resources	26
4.2	Workload = Work + Load	29
4.3	System and Subsystem Performance Specification	34
4.4	Abstract Virtual Machine	37
4.5	Module and Component Specification	37
4.6	Data Model	39
4.7	Typing of Operations	41
4.8	Chapter Summary	44
5	Performance Engineering of Information Systems	47
5.1	Non-functional Requirements	47
5.2	Lifecycle Models for Development of Information Systems	48
5.3	Motivation for Performance Engineering	54
5.4	Quantitative Performance Modelling	57
5.5	Qualitative Performance Modelling	67
5.6	Chapter Summary	67
6	Method for Performance Engineering of Information Systems	69

6.1	Method Overview	71
6.2	Worlds	82
6.3	Rationality of Design Process	85
6.4	Object-oriented Performance Engineering?	86
6.5	Comparison with Current Best Practice	88
6.6	Chapter Summary	90
7	Application to a Transaction-oriented System	91
7.1	Specify System Requirements	92
7.2	Establish Components: Projected Application	97
7.3	Establish Components: System Platform	100
7.4	Evaluate and Validate Static Model	104
7.5	Summary of Findings	106
8	Organisational Change and Workflow Technology	109
8.1	Organisational Change	109
8.2	Workflow and Workflow Systems	118
8.3	Chapter Summary	127
9	Extending the Method to Workflow Systems	129
9.1	SP as Organisational Models	129
9.2	Basic Framework	130
9.3	Taxonomy of Degree of Determinism	132
9.4	Differences between Human and Computerised Resources	135
9.5	Similarities between Human and Computerised Resources	135
9.6	Generic SP Model of Workflow Systems?	137
9.7	Chapter Summary	138
10	Application to a Workflow Organisation	139
10.1	Specify System Requirements	140
10.2	Establish Components: Telex Secretary	147
10.3	Evaluate and Validate Static Model of Telex Secretary	156
10.4	Establish Components: Contract Specialists	163
10.5	Evaluate and Validate Static Model of Contract Specialist	164
10.6	Summary of Findings	165
11	Application to a Workflow Computer System	169
11.1	Specify System Requirements	170
11.2	Establish Components: Lotus Notes Client-server	173
11.3	Establish Components: Replication Architecture	176
11.4	Establish Components: Customising Lotus Notes	177
11.5	Establish Components: Secretary Using Lotus Notes	179
11.6	Evaluate and Validate Static model for Computerised Solution	180
11.7	Similarities Between Manual and Computerised Solutions	188
11.8	Generalisation of the Case Study	189
11.9	Summary of Findings	191
12	Conclusions and Further Work	193
12.1	Major Contributions	193

12.2 Limitations and Further Work	195
Appendices	197
A Replication in Statoil	199
A.1 Replication Plan	199
A.2 Network Service Classes	201
A.3 Email Transport	202
B The Experimental CASE Tool PPP	203
B.1 PPP Languages	203
B.2 PPP CASE Tool	210
List of Figures	213
List of Tables	217
Bibliography	219
Index	235

Chapter 1

Introduction

An information system must have acceptable *performance* in terms of response time, throughput, and utilisation of the available resources. *Performance engineering* of information systems aims at developing systems with acceptable performance. **The overall research problem of this thesis is the feasibility of performance engineering for computer supported information systems within human organisations.** Contemporary approaches to performance engineering focus on dynamic models of resource contention and process synchronisation typically represented by networks of queues, state transitions or process interactions. These dynamic models may be solved by a variety of well-established analytic and simulation techniques. The difficulties which arise for large-scale problems are twofold:

1. Estimating appropriate parameters which capture the properties of the software and the organisation.
2. Maintaining an overview of a complex model, which is essential both to guide the choice of parameters and to ensure that the performance engineering process is an integral part of the wider system development process.

1.1 Research Context

This work was initiated during the IMSE project.¹ As part of the IMSE project, Andreas Lothe Opdahl, Vidar Vetland and myself applied SP to performance engineering of information systems [BOVS92]. We found that the static performance mod-

¹The IMSE project aimed at developing an integrated performance modelling support environment. It was a collaborative research project supported by the Commission of the European Communities as ESPRIT II project no 2143, and was carried out by the following organisations: BNR Europe STL, Thomson CSF, Simulog A.S., University of Edinburgh, INRIA, IPK (Berlin), University of Dortmund, University of Pavia, SINTEF (University of Trondheim), University of Turin and University of Milan. The project started in January 1989 and was finished in December 1991.

elling method SP (Structure and Performance) developed by Peter Hughes [Hug88] provides a suitable bridge between modern CASE tools and the traditional dynamic methods, in particular because it addresses the problems of parameterisation and overview identified above. Three thesis themes emerged from our participation in the IMSE project (Figure 1.1). Opdahl defended his thesis in 1992 [Opd92] and Vetland in 1993 [Vet93]. Thus, I am the last in a group of three doctor students who have been working with performance engineering of information systems in the Information Systems group.

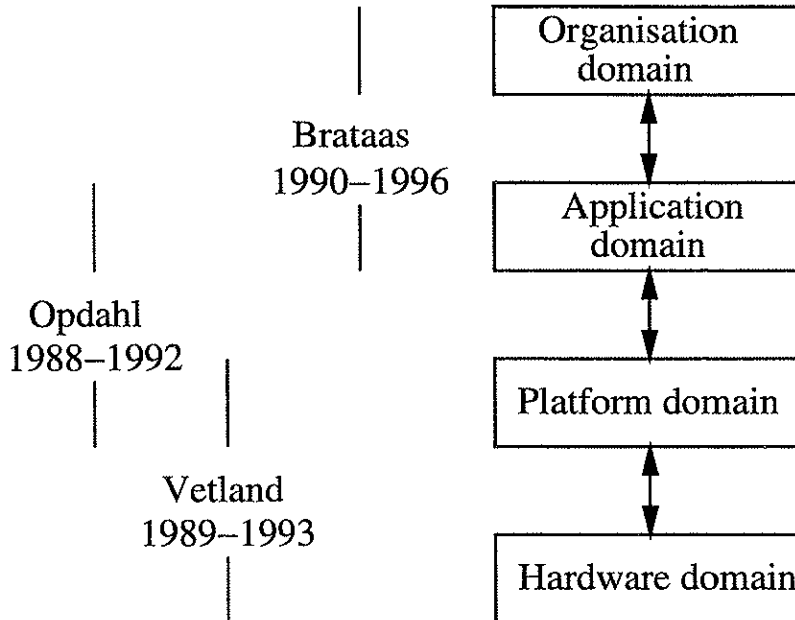


Figure 1.1: Schematic relation between three performance engineering theses in the Information Systems Group.

1.2 Investigating the Problem

To get practical experience, the *Blood Bank Case Study* was investigated from the spring of 1991 until spring 1992. This was done as part of the IMSE project. The Blood Bank Case Study comprised performance engineering of a transaction-oriented information system which was developed for the Regional Hospital in Trondheim. Process models were specified in the experimental CASE tool PPP and were annotated with parameters describing estimated use of software resources, showing the practical feasibility of a method for using performance engineering tools integrated with common CASE tools [BOVS92].

While there were no performance problems in the computerised parts of the blood bank system, severe problems were discovered in the interaction between humans and computers. Therefore, we introduced human resources into the framework, and extended our scope to encompass workflow systems. *Workflow systems* are information systems for supporting execution of *organisational processes* (workflows). In workflow systems, there is close interaction between *computational resources* and *organisational resources*.

The *Gas Sales Telex Administration Case Study* in the Statoil oil company was initiated during the autumn of 1993. Human resources were included in the modelling framework. This case study was finished during the spring of 1995. The Gas Sales Telex Administration Case Study looks at performance engineering of a workflow system for telex handling. The original Statoil project was initiated because the old manual information system did not offer acceptable response time and throughput. The case study consists of the performance modelling of human activity in interaction with a computer platform containing Lotus Notes.

In order to support the Gas Administration Case Study we had to develop a performance model of Lotus Notes. This model was based on measurements of external behavioural properties. The model had to be developed without deep knowledge of the internal structural properties of Lotus Notes. The prediction accuracy of the Lotus Notes model suffers accordingly. We have not made any efforts to develop the Lotus Notes model beyond what was necessary to show the feasibility of the proposed modelling framework.

1.3 Research Objectives

The major problems addressed in this thesis are:

1. How is it possible to estimate the *performance of workflow systems* prior to their realisation to ensure that they satisfy stated *performance requirements*?
2. Is it possible to treat the performance aspects of the (human) organisation and the computer system within the same framework?

For each of these research problem, a corresponding research objective is formulated:

1. Describe a method for performance engineering of workflow systems based on SP.
2. Investigate to which extent human resources can be modelled like computer resources in SP. Find the equivalences and the differences.

1.4 Claimed Contributions

This thesis contributes to the state of the art as follows. Concerning the method, the major contribution is:

- A method for performance engineering of information systems is proposed, based on SP. This is a step forward with respect to the state of the art in the field of performance engineering of information systems. The method is explored and illustrated with two case studies, the first focusing on a transaction-oriented information system, and the second on workflow systems and the organisation around a workflow system.

Concerning the similarities between computerised and human processes and resources, the major contribution is:

- An overall resource framework which integrates both human and computerised processes and resources has been developed. With respect to information processing, the organisational structure is analogous with the structure of a computer system. This framework has been tested out in the Gas Sales Telex Administration Case Study. Integration of human and computerised performance is particularly useful for workflow systems where the humans and the computer systems are heavily intertwined.

In this thesis, we have concentrated solely on bringing together methods for performance modelling of software resources and of human resources into one common framework. We have not made efforts to improve the state-of-the art of performance modelling of software components beyond what was done in Opdahl's and Vetland's contributions.

Chapter 2

Basic Framework

This chapter introduces the basic framework of the thesis. The framework has been developed to bridge the gap between performance engineering of computerised information systems and of organisations. The approach is motivated through the presentation of the Blood Bank Case Study.

2.1 A First Case Study — The Blood Bank

The motivation for trying to develop a common approach to deal with human and computer resources within the same framework stems from the Blood Bank Case Study as mentioned in the preface. Figure 2.1 depicts a simplified view of the blood bank using the PrM method in Section B.1.2. The basic human roles in the blood bank are donors, patients and laboratory engineers. The blood bank consists of the four main processes, donor administration, blood stock administration, transfusion administration and laboratory administration:

Administrate Donors Assisted by laboratory engineers, donors give raw blood which are stored in the blood bank. The blood bank is the name of the cabinet where all the different blood products are stored.

Administrate Blood Stock Makes sure the stock of blood will meet future demands for blood. Blood is also purchased from other blood banks.

Transfusion After testing, blood products are transfused to patients.

Laboratory Performs several tests on blood. Raw blood consists of several blood products which are grouped according to the ABO and the Rh test system. These test systems reduce the risk of giving patients blood products which do not match the patients blood type, which would usually have fatal consequences.

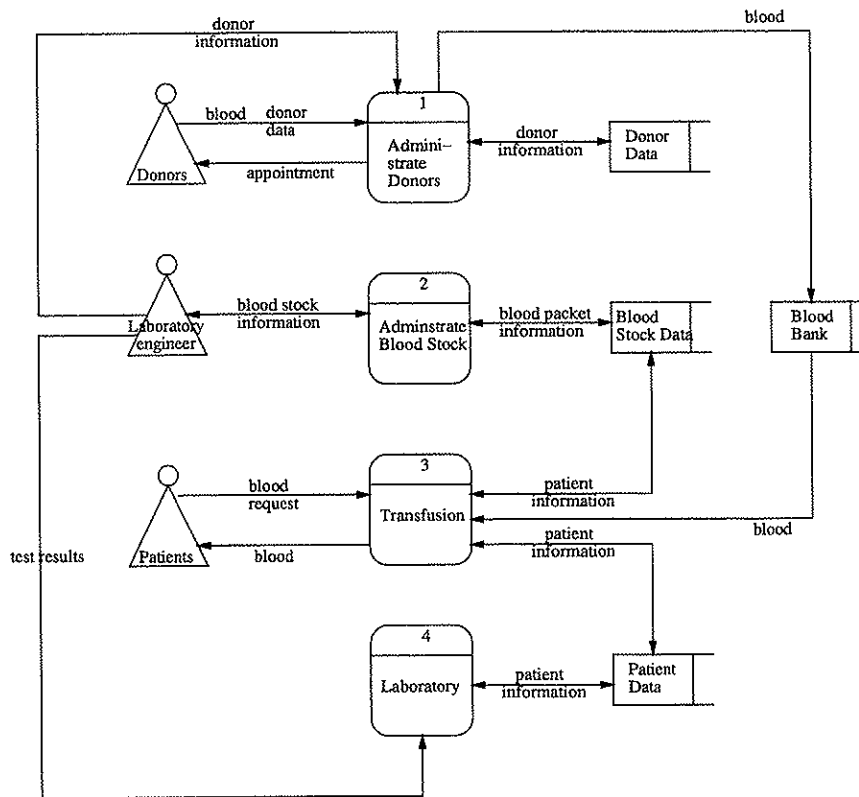


Figure 2.1: Top level blood bank model. These four information subsystems can be decomposed into 34 processes [Flø91].

2.1.1 The Original Manual Transfusion Process

Figure 2.2 shows informally the main components of the transfusion process, which take care of the transfusion of blood from the blood bank to patients in need for blood, and is a decomposition of process 3 in Figure 2.1. The processes of Figure 2.2 are carried out by humans. The relevant processes are:

Request for Blood Products The ward of the patient requests one or more blood products for the patient from the blood bank. This request is brought to the blood bank organisation on a sheet of paper.

Find Blood Product The blood bank organisation finds suitable blood products in the physical blood bank.

Physical Crossmatch of Blood Products Laboratory engineers mix a sample of donor blood with patient blood, looking for indications of mismatch.

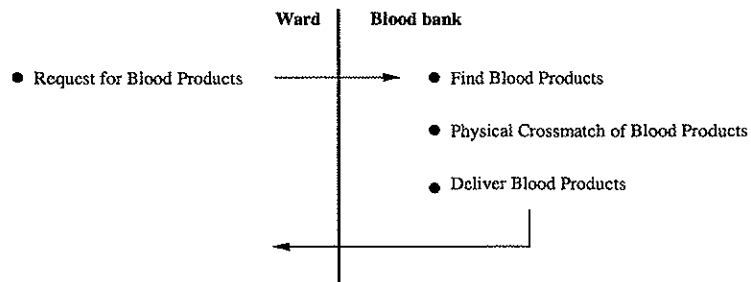


Figure 2.2: The manual transfusion process in the blood bank organisation.

Deliver Blood Products Blood products are delivered to the ward and the transfusion process is terminated as seen from the blood bank organisation's point of view (given that no complications arise).

2.1.2 The New Computerised Transfusion Process

It was decided to give computer support to the blood bank, and five computerised processes were added, which also brought about changes to the human processes. The computerised transfusion process is depicted in Figure 2.3. Process **Find Blood Products (3.3)**, **Physical Crossmatch of Blood Products (3.6)**, and **Deliver blood products (3.8)** are done by humans without computer support. The other process components are:

Fetch Patient Data (3.1) Patient data is fetched from the blood bank computerised information system.

Request Blood Products (3.2) Blood products are requested based on information from the ward. The information on the paper with the request from the ward must here be typed into the computerised information system.

Get Requisition Overview (3.4) A laboratory engineer investigates the requisitions made for a patient

Computerised Crossmatch of Blood Product (3.5) A laboratory engineer performs a computerised crossmatch between the donor blood and patient blood.

Register Result of Physical Crossmatch (3.7) Invoked for each blood packet requested for the patient.

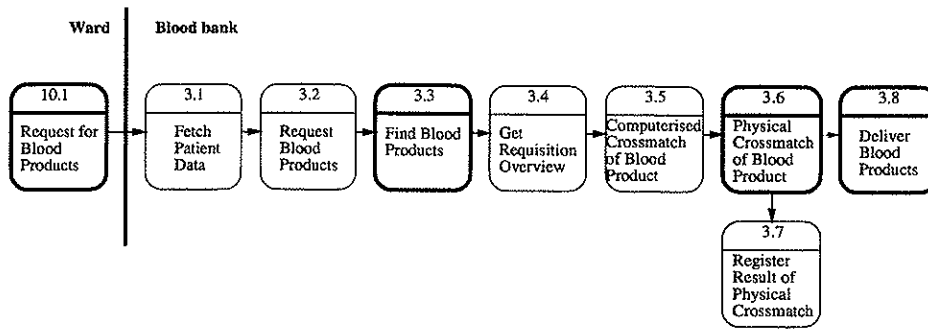


Figure 2.3: The transfusion process in the blood bank as it was implemented. Manual processes are indicated by **boldface** process symbols.

2.1.3 Performance Problems in the Manual Subsystem

The Blood Bank Case Study was done to investigate the modelling of computer performance problems. The study showed that there were no problems with the computer capacity, but after the system was put in operation, a severe performance problem was discovered in the interaction between the computer and the staff of the blood bank [BSH94, Bra94]. The introduction of the computerised information system actually slowed down the performance of the organisation. This experience changed the direction of this thesis into developing an approach for performance modelling that takes human aspects and computer aspects into consideration within the same modelling framework. To understand the problems with the transfusion process, this section first introduces some background material about both the manual and the computerised version of the transfusion process. A solution to the problems is then presented in Section 2.3 which is based on the performance engineering method of this thesis, introduced in Section 2.2. The problem processes are:

Get Requisition Overview (3.4) The computerised process took approximately 1 minute to execute, and was therefore never used. Part of the reason for this long time was that the patient's name had to be reentered, even if it was already present in the system. The system developers had not properly understood the workflow, and they did therefore not discover that process 3.4 would succeed process 3.3, which makes reentering of data unpractical.

Computerised Crossmatch of Blood Product (3.5) This process was not performed as prescribed, because it was much more convenient to look directly into the physical blood bank than to type in the necessary data and wait for the answer from the computerised information system. Instead, requests for blood packets were handwritten in a book, and fed into the computerised information system after the whole process had been terminated. Whenever the laboratory engineers wanted to look something up, this book was more convenient to use than the computerised information system.

Register Result of Physical Crossmatch (3.7) This process also created problems, because the process had to be performed once for every packet of blood which was requested for a patient. When up to 10 blood packets for one patient were needed in a hurry, this was not very practical, and created another argument for the informal book-based approach.

The basic problem in these three cases was the slow interaction between the manual and the computerised information system. If too much data had to be entered or if too many screen images had to be invoked, this would sometimes slow down the workflow and hence the performance of the organisation.

Accuracy of the information in an information system is important. Therefore, it is more critical if *update* operations are neglected, compared to *retrieve* operations. Thus, there are no serious problems if computerised retrieve operations like process **Get Requisition Overview (3.4)** and process **Computerised Crossmatch of Blood Product (3.5)** are not used, but replaced by some manual processes. However, this could have been discovered by including manual work for entering data in a performance model used during development of the computerised information system. But the process **Register Result of Physical Crossmatch (3.7)**, which *updates* information is more important. This process will therefore be considered more closely in Section 2.3.

The information system processes (i.e. the composite manual/computerised processes) were not explained in the information system documentation, and this made it very hard to get an overview of the relations between the different screen images which were used to enter and display information. It required careful study of the original documentation to find implicit information which could give some clues to understand the total information system processes. In addition, interviews had to be performed. If the information system processes had been recorded in the first place, understanding the transfusion process would have been easier.

2.2 Basic Framework

The basic framework introduced in this thesis views *organisations, humans, (platform) software* and *hardware* as a hierarchy of resources which are used by workflow processes in the *workflow system*. As illustrated in Figure 2.4, workflow processes use organisational resources which again use humans. A *workflow* is a process in an organisation. Workflow processes also use software resources which finally use hardware resources. *Actors* start workflows and use the result of workflow. The solid lines inside of each ellipse describe relationships between the objects in each ellipse, whereas the dotted lines describe relationships between objects in different ellipses. This framework is elaborated upon below. For more comments on this framework see Section 9.2. Figure 2.6 gives a rough indicating of the focus of each chapter in this thesis in terms of the overall framework in Figure 2.4.

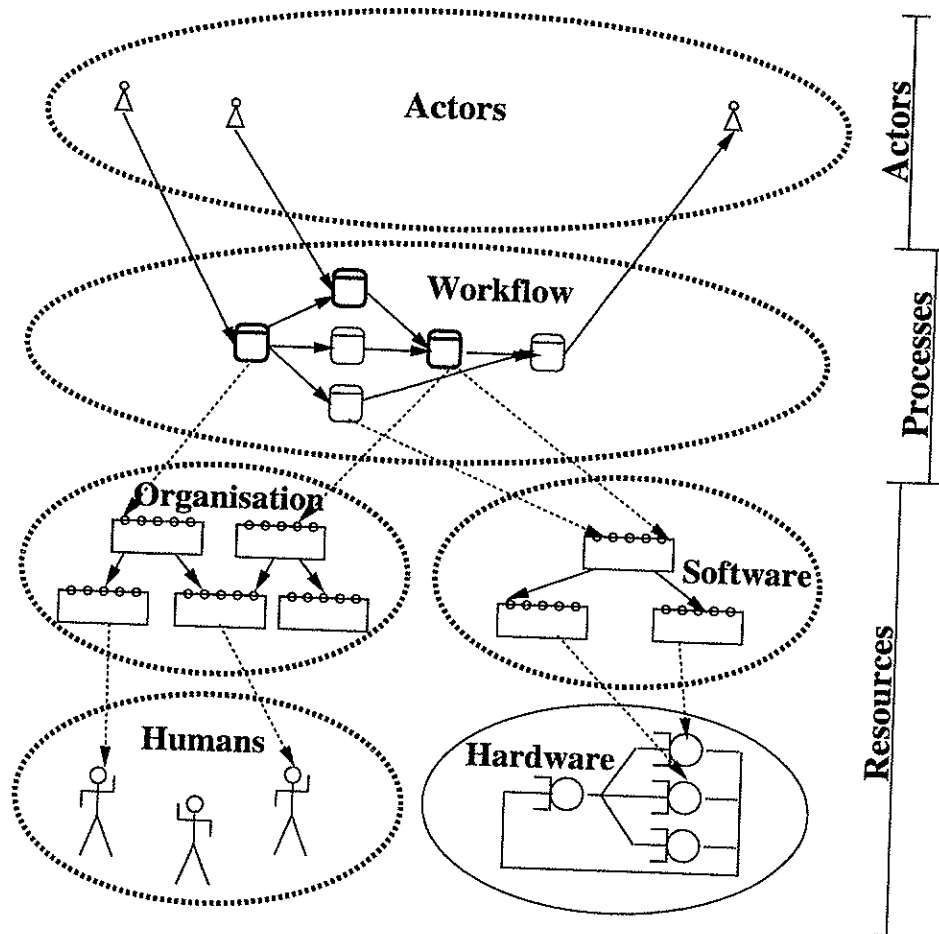


Figure 2.4: Basic framework for performance engineering in workflow organisations. The legend is in Figure 2.5.

Resources in the *organisational domain* consist of hierarchies of organisational units like departments and sectors which finally will devolve work on *humans*. Resources in the *workflow system* domain basically encapsulate workflows with both human and computerised parts, for example the workflow which was described earlier in this chapter. Typical examples of *software* resources are a database or a file system, while a disk, a network or a CPU are *hardware* resources. An information system¹ contains information processes which are executed with resources. A computerised information system uses only *computerised* resources. A computer system is another name for the resources which are used by a computerised information system.

¹This is not intended to be an extensive definition of an information system. A more complete definition is [FRI95]: "An *information system* is the conception of how information-oriented aspects of an organisation are *composed* (actors, resources, etc.) and how these *operate*, thus describing the (explicit and/or implicit) information-providing arrangements existing within that organisation." The terms typed in the sans serif font build on other definitions.



Figure 2.5: Legend for the basic framework in Figure 2.4.

Note that the term *resource* is used more narrowly than the ordinary use of this term. In this thesis, a resource acts reactively. A resource carries out work which is initiated by an actor. In one sense a resource is more passive than an actor, because the actor determines the goals of the work to be carried out. On the other hand, the resource does the actual work and may therefore be considered to be more active than the actor who merely waits for the results. Resources can either be *existing* and possible to measure or *projected* and impossible to measure. In either case, it is possible to model the resources. For existing resources measurements are used to construct these models. For projected resources, estimations are needed. Each resource offers some *operations* to resources at higher levels of abstraction and uses operations from resources at lower levels of abstraction. There is a difference between the *number* of operations and the *type* of operations: in this thesis, the term operation is used for the *type* of operation, each invocation of an operation is counted by the number of active customers.

Initially, I reacted against the idea of bringing humans into the same framework as the computer, because I felt this would reduce humans to the level of machines. However, the Blood Bank Case Study showed the need for viewing the two together. Early organisational theory also focused on the mechanistic view [Mor88], e.g. Taylor's scientific management. A comprehensive theory of organisational performance must also take other views into account, but this is outside the scope of this thesis. This thesis tries to develop a basic framework, which is needed prior to addressing more complex problems. For a more elaborated discussion on the relationship between humans and machines, see Chapter 8.

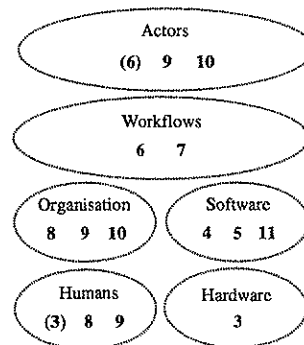


Figure 2.6: The structure of this thesis in terms of Figure 2.4.

2.3 Applying the Basic Framework on the Transfusion Process

The problem with the transfusion process described in Section 2.1.3 shows the tight coupling between a computerised information system and a manual information system. These systems do not use the same type of resource, but could they be considered under a common approach? This section will apply the basic framework in Section 2.2, and sketch a solution.

As described in Section 2.1.3, the process **Register Result of Physical Cross-match (3.7)** was critical. Bad performance in this process is particularly critical when several blood packets are needed in a hurry. The need for blood packets may be described by the operation `get_blood_product` in the blood bank application in Figure 2.7. This operation is indicated with a circle in the Blood bank application resource. For this `get_blood_product` operation, resource consumption for both the Blood bank computer system and the manual information system with the Laboratory engineer must be specified.

We will focus on the manual resource Laboratory engineer, because it was the human interaction (with the computer system) which was problematic, and not the computer system. For the manual operations of the Laboratory engineer, it will be sufficient to specify the operation `enter_data`, which for example may take 20 seconds. Thus, entering data 10 times, one time for each of the 10 packets of blood, will result in a response time of about $\frac{20}{60} \cdot 10 \approx 3$ minutes. If for example five patients need 10 packets each, this will take around 15 minutes, for the interaction only with the computer system. With such response times, we obviously have performance problems. Calculations like this are elaborated upon in the Gas Sales Telex Administration Case Study in Section 11.5.

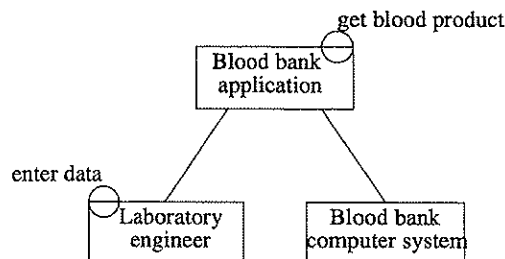


Figure 2.7: A sketch of the resources needed for earlier prediction of potential problems. The legend is in Figure 2.5.

2.4 Performance Measures in Different Domains

The term “performance” is used in a limited sense in the information system performance engineering domain and in this thesis, compared to the much broader use of the same term in the organisational domain. In this thesis, *performance* is defined as a quantitative property of a *system*, which can be measured in terms of: *throughput* or *response time*. Examples are: a throughput of 35 blood packets a day, or a response time of 1 minute for finding a suitable blood product in the blood bank. See Section 3.2.1 for other performance measures.

In the organisational domain, performance may also mean long-term performance i.e. how well overall organisational goals are met. For long-term measures, this thesis uses the term *effectiveness*. The term “efficiency” is also related to performance. Efficiency is the ratio of useful work performed relative to the total energy used [Hor92]. Thus, efficiency applies both to computer systems and to organisational systems. This section elaborates on the distinction between the terms performance, effectiveness, and efficiency as used in this thesis.

Effectiveness is related to *doing the right things*, in contrast to efficiency which focuses on *doing things right*. The effectiveness of a system is the degree to which the system meets longterm goals. In the computerised information system domain, examples of effectiveness measures are accuracy [Chu93], security [Chu93], availability, reliability, user-friendliness, and performance. Therefore, response time which is a performance measure, also is an effectiveness criteria.

In the organisational domain, all the above effectiveness measures are relevant. In addition, other effectiveness measures are also needed. Some authors view customer satisfaction as the ultimate effectiveness measure [CH92]. Customer satisfaction has to do with getting the right output at the right place, at the right time and at the right price [Har91]. Adaptability, i.e. the flexibility of the process to handle changes, may also be important for customer satisfaction.

Many organisations having customer satisfaction as part of their strategy do little to measure it [Ecc91]. Since what is measured receives attention, this is problematic, particularly when rewards are tied to the measures. As an example of missing measurements, each department in a manufacturing organisation may have optimised their part of the total process, without having measured the effectiveness of the overall process [DS90]. When measuring for example the response time for the total customer delivery process, this may be quite long. Methods for measuring organisational effectiveness with financial measures have been refined during the past four hundred years and are supported by vast institutional infrastructures [Ecc91]. This may mean that developing new measures will need substantial resources.

In most organisations, human effectiveness is supervised. However, the degree of formality will differ greatly from unofficial surveillance to formal computer surveillance. Computer surveillance of human effectiveness can be acceptable if certain constraints are adhered to [GH91]. It is for example important to keep the granularity of the surveillance so high that it gives the persons the possibility to be effective and not only efficient. In the struggle for efficiency, it is easy to lose sight of effectiveness [Cov90]. Improvements of organisational efficiency may for example lead to painful stress injuries [BG92], which is not very effective for the persons involved. As another example of unbalance between efficiency and effectiveness, increases in information system efficiency in an organisation may be taken out in terms of increased amount of information, improved quality of the information and better presentation, but not in increased effectiveness of the organisation [Sim94]. Again, this leads to suboptimal solutions.

Until now, this discussion has been on an organisational level. On a social level, the lack of empirical evidence for a correlation between computerised information system spending and productivity is termed the *productivity paradox* [Bry93, Str94]. Possible explanations of the productivity paradox are mismeasurements, lag before results show up, redistribution of profits within the organisation or mismanagement of information systems [Bry93].

2.5 Chapter Summary

This introductory chapter is suggesting that the interaction between organisational performance and computer system performance would benefit from modelling in the same framework. This has been supported by some problematic workflows in the blood bank at the Regional Hospital in Trondheim. The basic framework which is the main contribution of this thesis has been outlined. Concluding this chapter was a comparison between the concept of performance as used in this thesis and the standard concepts of effectiveness and efficiency. While effectiveness focuses on long term effects of a system or an organisation, performance as used in this thesis is a quantifiable property of a system which may be measured in terms of throughput and response time. Efficiency is related to doing things right, in contrast to effectiveness which focuses on doing the right things.

Chapter 3

Performance Modelling

The aim of *performance modelling* or *performance evaluation* is to estimate the performance of a system with a certain workload. *Performance models* which represent the performance, the workload and the system are an important part of performance modelling. Performance modelling is part of the wider concept of *performance engineering*, which is introduced in Chapter 5. This chapter briefly introduces some basic concepts of performance modelling. Since performance modelling traditionally is mostly used in computerised systems, this chapter will also focus on such systems. For more information, see textbooks like [Fer78, FSZ83, LZGS84, ABC88, MAD94] or the survey [HL84].

3.1 Workload, System and Performance

Both the workload W and the system S must be specified before it is meaningful to estimate the performance P of S as shown in Figure 3.1. This is also illustrated by the functional representation [Hug96]:

$$P(S, W)$$

Many disputes over performance could have been avoided if developers and users remembered to include not only S , but also W in the performance specification. The workload and the system are heavily intertwined. Depending on the boundaries

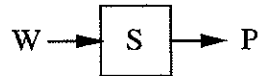


Figure 3.1: W gives workload on the system, S , with the resulting performance P .

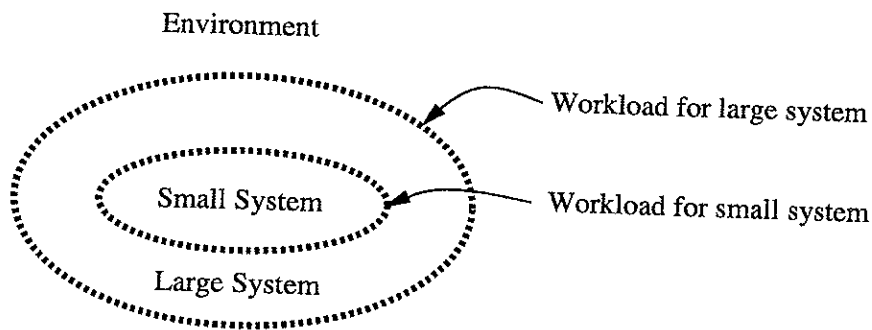


Figure 3.2: Both the small system and the large system have a workload. The workload on the small system belongs to the large system.

between the system and the environment, part of the “system” could belong to the “workload” and vice versa. This is illustrated in Figure 3.2 where a small system is part of a larger system. The workload on the small system will also be a part of the large system. As an example, the small system may be a computer system and the large system an organisation including a computer system.

As an example of how the notation $P(S, W)$ can be used, consider these definitions of design, improvement and comparison in the context of performance [Hug96]:

Design Find S such that:

$$P(S, W) \geq p_0$$

where p_0 is the performance requirement. Sizing is an example of design, and is to supply an information system with sufficient data storage and information processing power for a given workload.

Improvement Given S and W , find a modified system S' such that:

$$P(S', W) > P(S, W)$$

Comparison Given a workload W and systems $S_1, S_2 \dots S_n$, find the ranking or ratios:

$$P(S_1, W) : P(S_2, W) : \dots : P(S_n, W)$$

3.1.1 Performance Characterisation

Response time and throughput are the most commonly used performance measures. Response time is measured in time units, e.g. 1 day or 2 microseconds. Throughput

is measured in terms of departures per time unit, for instance 10 telexes per day or 100 disk accesses per second. All performance measures are interrelated. For example, for a system in equilibrium, throughput (X) and response time (R) are related by Little's law:

$$N = X \cdot R$$

where N is the number of customers in the system. As a special case of Little's law, the throughput X of a system and the utilisation U of the resources inside of the system are related by the equation $U = XD$, where D is the resource demand for each customer. See also Section 3.2.1 for other measurable quantities of a system. Other relations between measurable quantities are shown in [LZGS84].

3.1.2 Workload Characterisation

Ferrari defines a workload as the set of all inputs (programs, data, commands) which the system receive from its environment during a certain time frame [Fer78, p. 221]. This view will be expanded in more detail in Chapter 4, in the context of the SP method. For more information on workload characterisation, consult the standard textbooks, e.g. [Fer78, FSZ83] where useful concepts are defined, e.g. two workloads for the same system are equal if their performance is equal [Fer78]. [CS93] gives a survey of workload characterisation methods for hardware.

Two obstacles in workload characterisation are the two feedback loops in Figure 3.3, namely the feedback from the environment because of the performance, and the feedback from the system to the workload [Fer78, p. 234]. The feedback from the environment may either result in more workload or less workload on the system. For a telephone switching system, people will often call repeatedly if they do not get through, hence the workload will increase. Increased response time may also result in less workload on the system, because users may react by not using the system. This feedback loop was considered for the Blood Bank Case Study in Chapter 2.

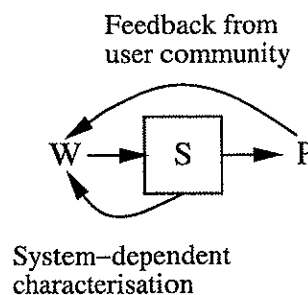


Figure 3.3: Two feedback loops causing problems for workload characterisation.

The cause of the feedback from the system to the workload is system dependencies in the workload. System dependencies prevent reuse of a system specification with another workload or prevents the reuse of a workload specification with another system. The key to system independence is to construct the workload specification in terms of logical resources instead of physical resources [Fer78].

3.1.3 System Characterisation

For performance modelling, the system S must be *complete*, meaning that it must include all the resources used by the system. This use of the term *system* differs from the use of the same term in information systems, where it is easy to forget that the hardware resources also are part of the system, as shown in Figure 3.4. Performance is holistic and depends on overview of all the resources which are directly or indirectly used by the workload.

For a *closed system*, the number of customers inside of the system is constant. For an open system, the number of customers inside of the system is not fixed, consequently customers are crossing the system boundary in this case.

An important part of system characterisation is contention modelling and instrumentation. Contention modelling is described in Section 3.2. For information on instrumentation, see [Fer78]. Tools for program instrumentation exist [Amm92], which could guide implementation decisions, like selection of a suitable storage structure for an editor [AB88] or a sorting algorithm [AWS91].

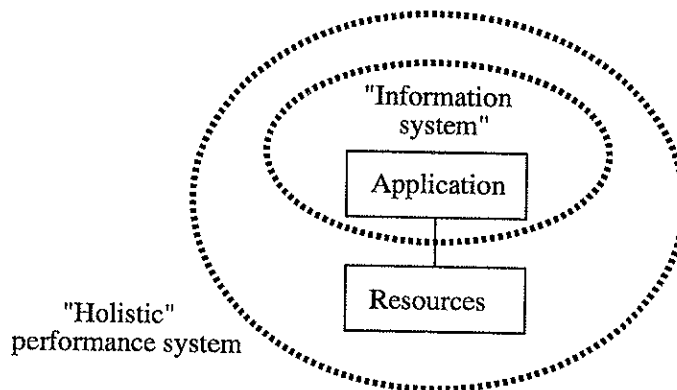


Figure 3.4: An "information system" which does not take (hardware) resources into account in contrast to a "holistic" performance system where all the resources are included in the system.

3.2 Contention Modelling

In cases where resources are limited, i.e. where there is a queue for a given resource, then the processes using this resource will contend for resources while waiting in the queue. While a dynamic performance model represents contention, a static performance model does not. Static modelling is considered in Chapter 4. As shown below, the two main modelling paradigms for dynamic contention modelling are the queue-oriented and the state-oriented paradigm:

Modelling paradigm	Solution method	
	Simulation	Analytic
Queue oriented	x	x
State oriented	x	x

As shown, both the queue-oriented and the state-oriented paradigm may be solved analytically or with simulation. Complex models in both paradigms will often require simulation, e.g. in general it is not possible to analytically solve a queueing network model with priorities between several classes. SIMULA is an example of a (Norwegian!) simulation language [Bir79]. Solving models with simulation will usually use more computer resources than solving the same models analytically. On the other hand, complex analytic models may require costly numerical techniques for getting sufficient accuracy.

Queueing networks and bounds analysis are examples of a queue-oriented modelling paradigms, which are described in Section 3.2.1 and 3.2.2, respectively. Examples of a state-oriented modelling paradigm are Petri nets and Markov chains. Petri nets are considered in Section 3.2.3. For a description of Markov chains, see for example [TK84]. Apart from the state-oriented and queue-oriented paradigm, we have the process-oriented paradigm, which can only be solved by simulation, not analytically. PIT (Process Interaction Tool) is an example of the process oriented paradigm [Bar89, BH90]. PIT models generate SIMULA code.

3.2.1 Queueing Networks

This section gives some background material for operational modelling of queueing networks. The operational approach to performance modelling of queueing networks was a breakthrough in terms of simplicity compared to the stochastic approach which was previously used. While the stochastic approach uses distributions for each parameter, the operational approach uses average numbers as parameters, thereby reducing the parameter capture effort. At the same time, the accuracy of the operational approach is for several problems comparable to the accuracy of the more complicated stochastic approach. The operational approach to bounds analysis are introduced in Section 3.2.2. This presentation is based on the textbook [LZGS84], where more details can be found.

The service centre is the basic element of a queueing network. Each service centre represents some resource in a system and has an average service time for each class using the resource. A class represents a group of customers who use the centre in the same way. Hence, a single class queueing network will only represent one class, whereas a multi class queueing network may represent several classes. There are two types of service centres:

Delay centres where there is no contention. Each customer visiting the centre has been logically allocated their own resource, so there are no queues. The most common use of a delay centre is to model the think time of terminal users.

Queue centres with contention for resources, meaning the customer has to wait for service at this queue centre in a queue. Disks and CPUs are typical examples of queue centres.

Figure 3.5 depicts an open queueing network with four queueing centres. The basic relation for open queueing networks is the equation:

$$R_k = \frac{D_k}{1 - U_k}$$

The equation above states that the residence time R_k for a queue centre k is proportional to the resource demand D_k for this service centre, and inversely related to the fraction of this centre which is unused ($1 - U_k$). For a delay centre where there is no queue, this relation is simply $R_k = D_k$. Thus, the factor $\frac{1}{1 - U_k}$ accounts for the queue in a queue centre. As an example, if a queue centre has an utilisation of 50 %, i.e. $U = 0.5$, $R_k = 2D_k$ for this queue centre. A 40 % increase in the utilisation to 70 % will give a residence time $R_k = 3.3D_k$ which is 67 % higher than the original residence time of $R_k = 2D_k$. For utilisations over 50 %, residence times are no longer roughly linear with respect to utilisation as shown in Figure 3.6. Closed queueing networks have more complex equations.

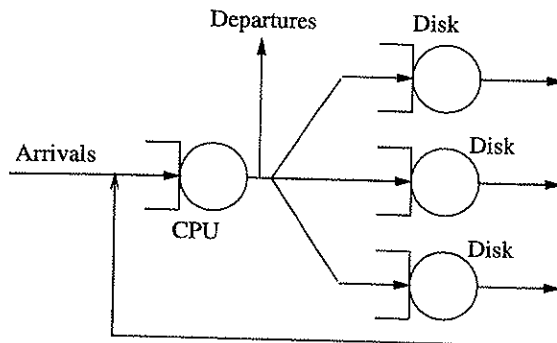


Figure 3.5: An example of a simple open queueing network. This queueing network represents a CPU and three disks as a queue centres.

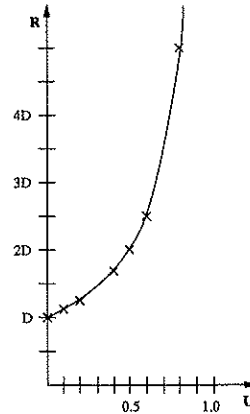


Figure 3.6: A graph of the relation $R = \frac{D}{1-U}$.

All the measurable quantities of a queueing network in the operational approach are:

A_k	number of arrivals observed	B_k	busy time
C_k	number of completions observed	D_k	service demand
N	customer population	R_k	residence time
S_k	service requirement per visit	T	length of observation interval
U_k	utilisation	V_k	number of visits
Z	think time of a terminal user	X_k	throughput
λ_k	arrival rate		

The subscript k refers to a service centre. Note that R_k is *residence* time for a single service centre k , while R is *response* time for a complete system. Also note that many of these performance measures could be used to specify performance requirement, even though throughput and response time are the most commonly used ways of specifying performance requirements.

Queueing networks which satisfy certain properties may be aggregated by using FESCs (Flow Equivalent Service Centre). With FESCs, it is possible to abstract away detailed parts of a queueing network, thus increasing the overview. Application of FESCs may also reduce considerably the cost of solving a performance model. On the other hand, by combining different FESCs with other solution approaches like for example simulation, it is also possible to increase the *accuracy* of performance models at a reasonable cost.

3.2.2 Bounds Analysis

A simple way of predicting performance for a *closed* queueing system is asymptotic bounds analysis [LZGS84]. A simple performance measure in asymptotic bounds

analysis is the intersection between optimistic light load and optimistic heavy load:

$$N^* = \frac{D + Z}{D_{max}}$$

where:

- N^* Number of terminals possible to support with acceptable performance.
- D Total service demand for all the resources in the system for one visit.
- D_{max} Service demand for the queue centre with the highest service demand (the bottleneck queue centre).
- Z Total average think time between each customer visiting the system.

3.2.3 Petri Nets

Petri nets [ABC88] consist of places and transitions as in Figure 3.7 where places are depicted as circles and transitions as lines. In a standard Petri net, transitions are fired when all its input places leading to a transition contain at least one token. Time in Petri nets can be introduced in two ways: (1) a duration between enabling and firing of a transition; (2) a token becomes available only after a duration in a place. This duration may be a fixed delay or a stochastic distribution. The exponential distribution is simplest to handle mathematically, since a Petri net with only exponentially distributed transitions is isomorphic to a Markov chain [ABC88]. Therefore, solving such Petri nets are equivalent to solving Markov chains. Because of a large state space even for quite small problems, this may be time-consuming, both if the chain is solved analytically or if simulation is used. In contrast to queueing networks, there is no support for abstraction in normal Petri nets. Extensions of Petri net with abstraction support, so-called behaviour net models (BNMs) are described in [SK93].

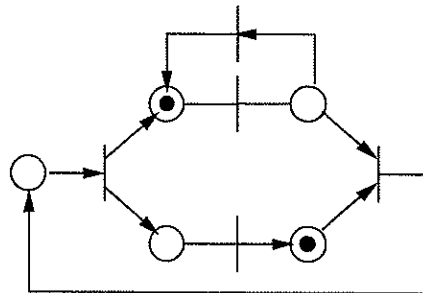


Figure 3.7: A simple Petri net.

3.3 Performance Modelling Cycle

The modelling cycle in Figure 3.8 shows the relation between an existing system, workload, performance, modified (or projected) system and the performance model. As depicted in the left part of this figure, the modelling cycle consists of three basic phases:

Validation phase: In this phase, a performance model is developed and parameterised. The system is measured to give the workload measure for the performance model. During the validation phase, the estimated performance of the performance model is compared with the *measured* performance of the *system*, under the same workload. To gain confidence in the performance model, this validation should be repeated for changes, both in the system and in the workload.

Projection phase: The performance model is now used to project the performance of a projected, modified system with a modified workload.

Verification phase: After the modified system has been finished, it can be measured. During the verification phase, the projected workload model is compared with the measured, modified workload, and the projected model is compared with the measured, modified system.

Note that the terms validation and verification as defined here with respect to performance modelling differ from the general definitions of the same concepts for information systems as described in for example [SK93].

3.4 Chapter Summary

The basic components in a performance model are a model of the system, a workload model and a performance model. The operational approach with focus on average numbers instead of distributions in the stochastic approach was chosen because of simplicity. In the operational approach, analytic solution of queueing networks is possible. In the performance modelling cycle, the distinction between validation and verification is important. Model validation is part of model formulation, while verification is done after the model has been used to project performance for a projected system. Whereas the workload in validation can be derived from the existing system, the workload used in verification must be based on some kind of extrapolation of the present workload to cope with anticipated changes in the projected workload.

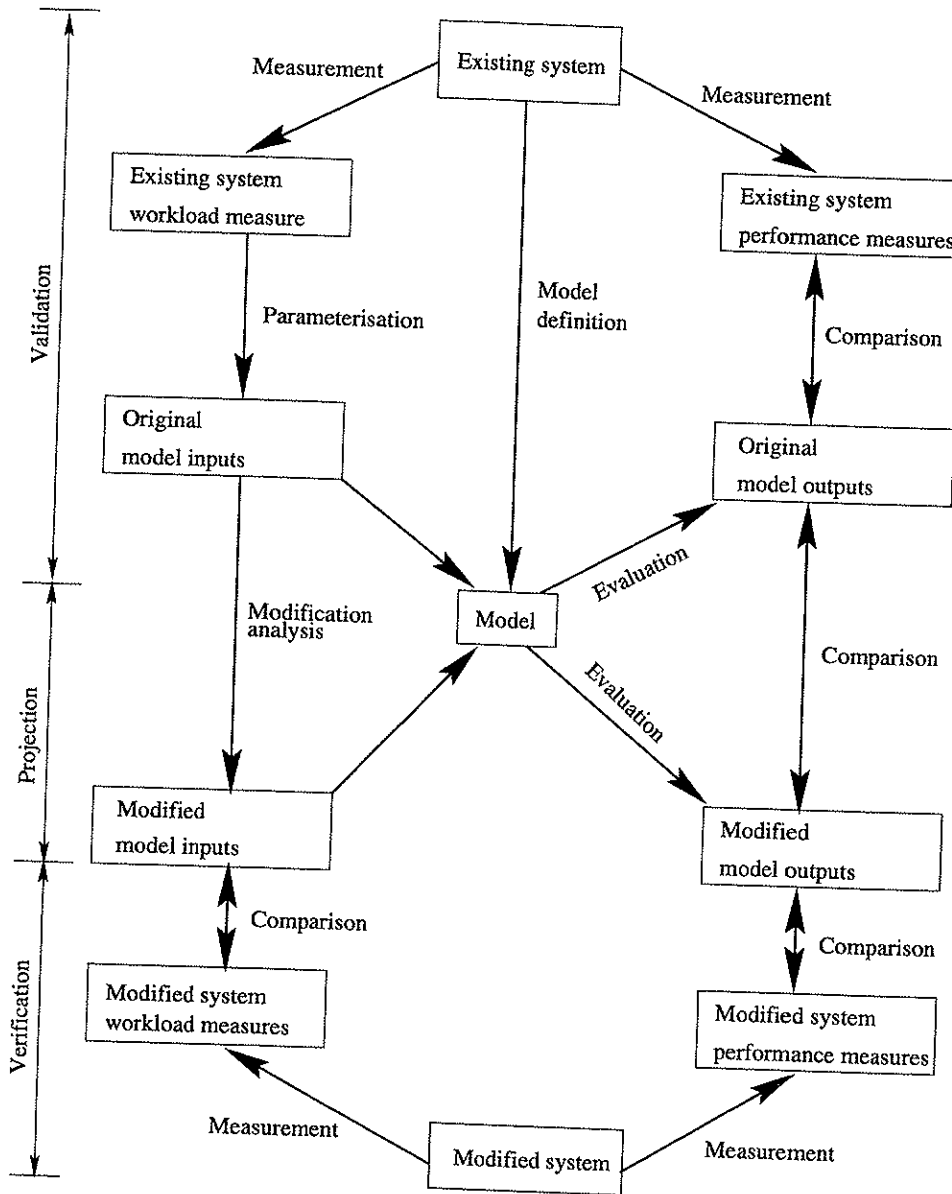


Figure 3.8: The performance modelling cycle [LZGS84].

Chapter 4

Structure and Performance (SP)

This chapter explains the basic concepts of the Structure and Performance (SP) method as used in this thesis. The material in this chapter found its form after extensive discussion with Peter Hughes, the originator of SP. SP is a method for describing the hierarchical and modular nature of a system and for analysing the resource requirements of its component parts [Hug93]. SP structures the interconnection between components such that two modellers who model the same artifact will tend to make equal models [Vet93, p. 228].

Until now, SP has almost exclusively been used to model computerised information systems. A tool supporting SP has been implemented [Min90] and SP ideas have strongly influenced Vetland's thesis [Vet93]. His thesis also describes a method for using SP in existing systems, including real-world examples of SP models. The best reference for the conceptual underpinning of the SP method is [Hug88], but see also [HBP⁺88, Hug89, Xen91, Hug93] for supplementary information.

This chapter is structured as follows: First, Section 4.1 introduces the concept of resource hierarchies which is central to SP. Then the separation between work and load in SP is explained in Section 4.2. Performance specifications for systems and subsystems are described in Section 4.3. The abstract virtual machine is a basic concept underlying the theoretical understanding of SP which is introduced in Section 4.4. SP modules may be extended to SP components as explained in Section 4.5. Section 4.6 describes SP's data model. Typing of operations is explained next, before these typed operations are used to form distributed and non-distributed architectures in Section 4.7. SP components and operations are written with the `teletype` font for clarity.

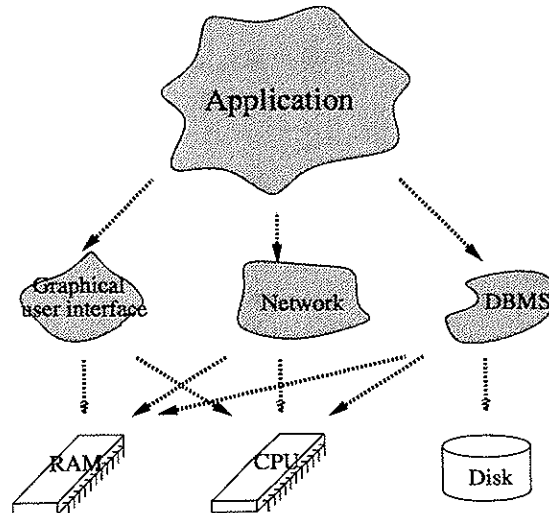


Figure 4.1: Type of resources used by an application. At the bottom, the resources are physical and concrete like memory, CPU or disk. Higher in the resource hierarchy, the resources are abstract. Not visible in this figure is the organisation which uses the application.

4.1 Hierarchies of Resources

Central to SP are *hierarchies* of resources. Figure 4.1 shows a typical hierarchy for a transaction-oriented application. This application gives workload on a graphical user interface, a network and a database. In turn, these resources will use primary memory, CPU and disk. In this thesis, the focus is on the time resources are used, e.g. the CPU usage and the disk usage. This is measured in the number of instructions during a given time interval, the *workload*. Workload will be separated in *work* and *load* later in this chapter. This thesis will especially focus on *work*.

Use of primary and secondary memory resources in the system may be crucial for the performance of the system. Even if memory becomes cheaper, it will still be a limited resource. During application development, it is for example easy to put too much data into primary memory which may lead to problems later during development, because slow, secondary memory must be used. As an example of memory considerations during design, client-server storage systems are analysed in [DM92]. While the primary focus of this thesis is on work, problems concerning *space* will also be touched upon.

The primary reason for including a resource in an SP model is to prepare for a *change* in this resource. SP makes a *conceptual* hierarchy of resources, not a physical hierarchy: thus there will not be several clients in a client-server model (See Section 4.7.2 for a client-server SP model.). If two clients are different, this conceptual difference

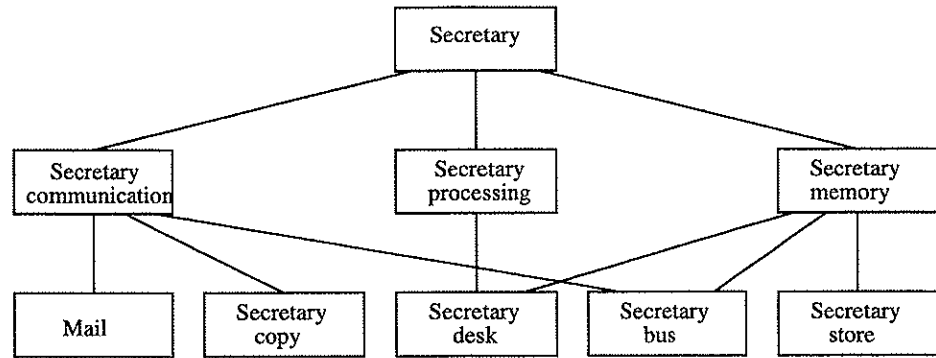


Figure 4.2: SP model for a secretary.

must be represented explicitly in the conceptual SP model.

To illustrate SP in this chapter, we introduce a model of a secretary in Figure 4.2. The secretary in Figure 4.2 is seen to possess operations for communication, processing and memorising. The secretary acts as a communicator when receiving and sending documents, and is using the `mail`, `secretary_copy` and `secretary_bus` resources, where the `secretary_copy` and `secretary_bus` resources are resources internal to the secretary, e.g. it takes secretary time to copy a document. `Secretary_bus` is related to the walking in corridors between, e.g. the archive and the desk. The secretary also performs processing of documents, using the resource `secretary_desk` which is internal to the secretary. Furthermore, the secretary memory operations is related to the archiving of documents, e.g. a secretary stores and retrieves documents, and uses secretarial resources `secretary_store` to do so. This model of a secretary is used in a slightly expanded version in the Gas Sales Telex Administration Case Study in Chapter 10.

4.1.1 Relation to Basic Framework

The basic framework in this thesis was introduced in Figure 2.4 and will be explained in more detail in Figure 9.2. Since this chapter and the following chapters 5 and 6 focus mostly on computerised information systems, Figure 4.3 shows an outline of the resource framework for performance engineering of computerised information systems.

This chapter focuses on static (software) modelling, and also comments on the interaction between static software modelling and dynamic hardware modelling. Contention in software and hardware may be modelled with dynamic techniques from Chapter 3, as described in [Vet93].

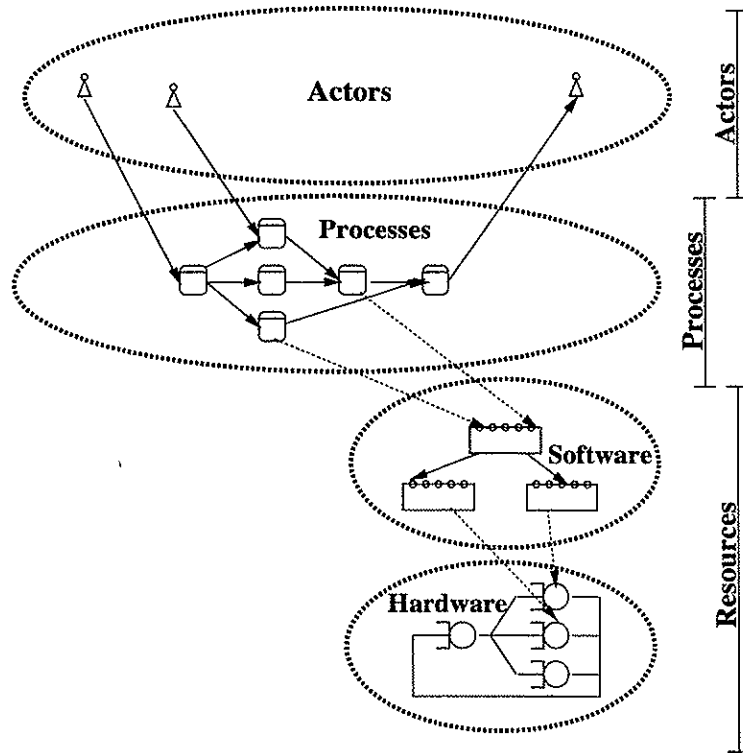


Figure 4.3: Framework for performance engineering of computerised information systems. The legend for this framework is shown in Figure 2.5 in Section 2.2.

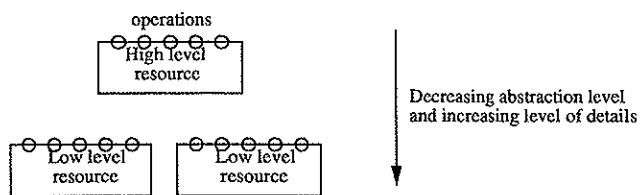


Figure 4.4: Work is always related to the use of operations at a given abstraction level.

4.2 Workload = Work + Load

Underlying the philosophy of SP is the distinction between work and load [Hug88]. Taken together, work and load define the workload for some system, illustrated by the symbolic equation **Workload = Work + Load** in the title of this section. An example illustrates the distinction between work and load: when you receive a document, the load applies to the point in time when you receive the documents during this period. At each point in time you receive a document, some organisational or computerised resources will be used. This resource usage is measured in terms of work. Work is always related to the use of *operations* from a resource at some abstraction level as illustrated in Figure 4.4. Operations at a high level of abstraction use other operations at a low abstraction level.

The workload for each operation will contain both the work and the load, i.e. both the amount of work for each operation, and the number of operations during a given time interval. Simply put, load is *when* an operation is invoked and work is *what* the operation does with the system. When a load model and a work model are combined with a contention model, we have a performance model [Vet93]:

$$\text{Performance Model} = \text{Load Model} + \text{Work Model} + \text{Contention Model}$$

Each of these aspects are described by a model, as illustrated in Figure 4.5. The load model is described in Section 4.2.2, the work model in Section 4.2.1 and the contention model in Section 3.2. As shown in Figure 4.5, the dynamic *load* model is only relevant in the contention model and not in the static *work* model.

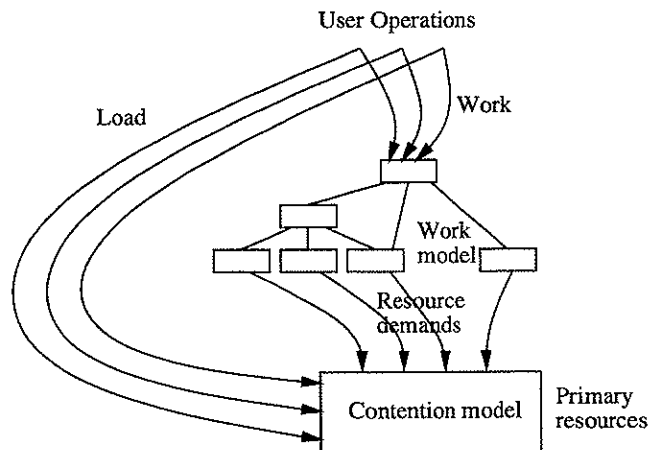


Figure 4.5: Combining a work model, a load model and a contention model gives a performance model [Vet93].

4.2.1 Work

Work is defined as a vector \vec{w} of operations applied on a given resource in SP.¹ Assume that a secretary handles documents, and performs an `in_doc` operation every time she receives a document. When she sends out a document she will perform an `out_doc` operation. Then the vector:

$$\vec{w}_{\text{Secretary}} = \begin{bmatrix} \text{in_doc} & \text{out_doc} \\ 2 & 1 \end{bmatrix}$$

states that the resource `Secretary` must perform 2 `in_doc` operations and 1 `out_doc` operations. For work, there is no information about when the operation occurs; how long time an operation takes and the sequence of operations. Thus, work is a static concept, since time is not involved. This is in contrast to the dynamic load concept where time is vital. But since things are not necessarily simple, a static work model may exhibit *load-dependence* as elaborated upon in Section 4.3.2.

The work vector \vec{w} may be normalised to a unit work vector \vec{u} which specifies the work for the “average” operation [Hug96]. If the total number of operations is v , \vec{u} is defined by $\vec{w} = v\vec{u}$, where \vec{u} have the elements x_i , $i = 1, 2, \dots$ and $\sum_i x_i = 1$. If for example: $\vec{w}_{\text{Secretary}}$, then $v = 3$ and $\vec{u} = [\frac{2}{3}, \frac{1}{3}]$. v is related to the visit count V by the equation: $v = b_{ss}V$ where b_{ss} is the batch size [Hug96].

Information work is related to physical work, not only conceptually [Kol86], but also in practice. Eventually, work in an information process will lead to changes of states in a hardware chip. Opdahl gives a survey and compares several definitions of computer work [Opd92].

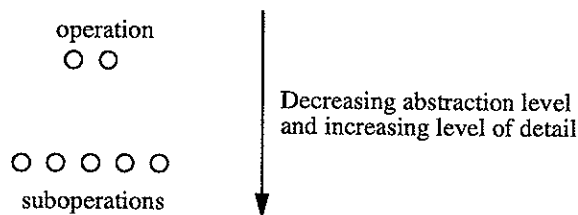


Figure 4.6: Work complexity is a matrix showing the relation between two levels of operation.

4.2.1.1 Work Complexity Specification

Work complexity is related to differences in work between two abstraction levels. In

¹More specifically, the vector of operations is applied on an abstract machine. This is introduced in Section 4.4.

Figure 4.6, the *work complexity specification* describes the relation between 2 high level operations and 5 low level operations. This work complexity specification is a matrix which has complexity functions for each element, as shown below where each element f_j^i in the complexity specification matrix $C_{Subsystem}^{System}$ is a complexity function:

$$C_{Subsystem}^{System} = \begin{array}{c} op_1 \\ \vdots \\ op_n \end{array} \begin{array}{c} subop_1 \dots subop_m \\ \left[\begin{array}{ccc} f_1^1 & \dots & f_m^1 \\ \vdots & \ddots & \vdots \\ f_1^n & \dots & f_m^n \end{array} \right] \end{array}$$

In this work complexity specification, **System** has n operations while **Subsystem** has m operations. For example in Figure 4.2 the complexity specification below represents the relation between the resource **Secretary** and the resource **Secretary.communication**:

$$C_{Secretary.communication}^{Secretary} = \begin{array}{c} in_doc \\ out_doc \end{array} \begin{array}{c} send_draft \quad fetch_post \quad send_post \\ \left[\begin{array}{ccc} 0 & 1 & 0 \\ x & 0 & 1 \end{array} \right] \end{array}$$

Each of the six elements in this matrix is a complexity function. The complexity function $f_{send_draft}^{out_doc} = x$ in this complexity specification describes the relation between the operation **out.doc** and the operation **send.draft**. The parameter x is a parameter which must get a value before this complexity function is determined. If $x = 0.5$, this means that for each **out.doc** operation, the operation **send.draft** is performed 0.5 times. This number may mean that on the average, only half of the documents which the secretary sends out needs a draft.

4.2.1.2 Problems with Work Characterisation

The basic problems with characterisation of work are [Hug96]:

Operational variety The system may offer a large number of operations.² In general, it may be impossible to characterise all these operations. For example, the complexity specification $C_{Secretary.proc}^{Secretary}$ above is a simplification. Several operations were not considered important, and consequently not modelled to cope with the problem of operational variety.

²The term *function* was originally used instead of *operation* i.e. functional variety [Hug78]. The term function is more general than the term operation. Commands in UNIX may for example have so many parameters that each function in reality hides several alternative operations. To keep the number of concepts low, this thesis uses the term operation.

Data dependence Performance may depend critically on the data which are operated on. The complexity specification $C_{Secretary_proc}^{Secretary}$ above, for example, depends on the data x . If either the number of data values is too large, or the volume of data is too large (e.g. an operation sorting 1000 numbers), some techniques for finding relevant combinations of operations may be used (see end of this section for more information).

Sequence dependence Performance may depend on the order in which the operations are invoked. This will only happen for state dependent systems. Software will be state dependent if synchronisation is involved, or even more subtle if for example database statistics may control the way the database looks for information. Hardware will be state dependent if performance depends on for example the position of a disk arm, or placement of files on a disk.

Theoretically, data dependence and operational variety are equivalent because for each permutation of data a new operation may be defined, leading to operational variety. For example, the walk operation in the complexity specification in Section 10.2.1 has the distance to walk as data, but each distance could also have been modelled with one operation, which may have led to the problem of operational variety. Operational variety is typically handled with cluster analysis while sequence dependence is handled by repeating the experiment for several sequences. Validation is important to justify the simplifications.

4.2.2 Load

In the ideal case, the load on an operation should represent each time the operation is used. This load description would be truly representative, but far from compact. Simplifications are used in practice. As stated in Section 3.2, this thesis focuses on the operational approach in contrast to the stochastic approach to load specification. Load in the operational approach may be modelled as *transactions* for an open system, and for a closed system load may be modelled with *batch* load or the more general *terminal* load [LZGS84]:

Transaction:	Intensity for one (unit) operation	λ
Batch:	Number of concurrent operations	N
Terminal:	Number of concurrent operations, and think time	$\{N, Z\}$

4.2.3 Workload

A *workload specification* will contain a description of both *work* and *load*. How work and load are combined to form workload is not straightforward. In the general case, a *system* will have n *operations*. Each operation represents one way of using

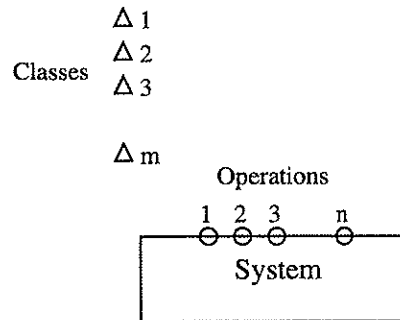


Figure 4.7: Each *class* represents one way of using the *operations* of a *system*.

the resources *inside* of the system. The *environment* has m ways of using these operations. Each way of using the operations is termed a *class*. This is illustrated in Figure 4.7, where the m classes are depicted as triangles and the n operations are illustrated with circles. Each class has a certain *load*. For example, class 1 may have an average of 30 visits during each day, in a transaction type of load specification:

$$\lambda^{Class-1} = 30$$

How each class uses the operations of the system is specified by a *work* specification. Assuming $n = 2$ (2 *operations*), class 1 may use operation *in_doc* $\frac{2}{3}$ times and the operation *out_doc* $\frac{1}{3}$ times. This work specification is a *normalised* work vector $\vec{u}_{Secretary}$ as described in Section 4.2.1:

$$\vec{u}_{Secretary} = \begin{bmatrix} in_doc & out_doc \\ \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

For class 1, the workload specification $W_{Secretary}^{Class-1}$ is now:

$$W_{Secretary}^{Class-1} = \lambda^{Class-1} \cdot \vec{u}_{Secretary}$$

$$W_{Secretary}^{Class-1} = \begin{bmatrix} in_doc & out_doc \\ 20 & 10 \end{bmatrix}$$

Formally, the workload specification for an open system of the transaction type with one class is:

$$W = (\lambda \cdot \vec{u}) = \vec{\lambda}'$$

This is the work which is devolved on some system, during a given time interval. With several classes, the vectors W , \vec{u} and $\vec{\lambda}'$ will be matrices and the scalar λ will be a vector. For a closed system with one class, the workload description is:

$$W = (N, Z, \vec{u})$$

N is the number of customers in the system. $Z = 0$ for a batch system. For a terminal system, $Z \neq 0$. \vec{u} specifies how each operation offered by the system is used by one class.

4.2.4 Resource Demands

The mapping between the work model and the contention model is specified by the *resource demand* model. Resource demands are specified as a vector. Below is an example of resource demands for the two operations of `Secretary_desk` in Figure 4.2:

$$D_{\text{Secretary_desk}} = \begin{matrix} \text{read} \\ \text{type} \end{matrix} \begin{bmatrix} 1 \\ 10 \end{bmatrix}$$

Whereas the time is irrelevant for a work specification, it is mandatory for a resource demand specification. Resource demands are always measured relative to some time interval. In this case, the time interval is 1 minute. Thus, it takes 1 minute to read one page and 10 minutes to type one page. In [LZGS84], the term service demand/requirement is used in the same way as resource demand in this thesis.

4.3 System and Subsystem Performance Specification

Before we can make a more general definition of a performance specification in Section 4.3.2, we must first define subsystems in relation to SP.

4.3.1 Subsystem

For hierarchical decomposition, a *subsystem* must include at least one physical resource. The subsystem must also be sufficiently separable from the rest of the system so that it is meaningful to define its performance quasi-independently. In practice, this means that the subsystem [Hug96]:

1. Shares some communication resource with the rest of the system.
2. Is otherwise self-contained. In particular, it does not share any of its resources with processes running outside of the subsystem.
3. Is separable with respect to the other subsystems in the system. A key requirement of separability is that the average rate at which the subsystem processes requests only depends upon the number n of the customers in the system [LZGS84]. In practice, separability is a question of approximation.

For a system, the first requirement is not necessary, but the second and third requirement are necessary.³ With these principles, it is possible to visually detect subsystems in SP models, e.g. principle 2 above is broken in Figure 4.2, since the operations for some of the components `Secretary_copy`, `Secretary_desk`, `Secretary_bus`, and `Secretary_store` are performed by the same physical resource, namely the secretary. This deficiency in the model is corrected in Chapter 10.

4.3.2 Performance Specification

A performance specification can be made for all subsystems which adhere to the restrictions in Section 4.3.1. The relation between work, load and workload is shown in Figure 4.8.

The different parts in Figure 4.8 can be represented as follows [Hug96, Hug95]:

Visit count:	V_s
Normalised unit work operation (Sec. 4.2.1):	\vec{u}_s
Work unit for system S :	(V_s, \vec{u}_s)
Processing concurrency:	n
Dynamic constraint (on the concurrency):	$n_{s,max}$
Processing rate:	μ_s
Complexity specifications are:	$\{C_k\}$
Devolved work-units:	$\{u_{ss}, \vec{u}_{ss}\}$.

³Note the slight clash between the common term “information system” and the term “system” as described in Section 3.1.3. An “information system” may not include the necessary software platform and hardware, and may therefore not be “self-contained”. Since the term “information system” is established, this thesis will still use this term.

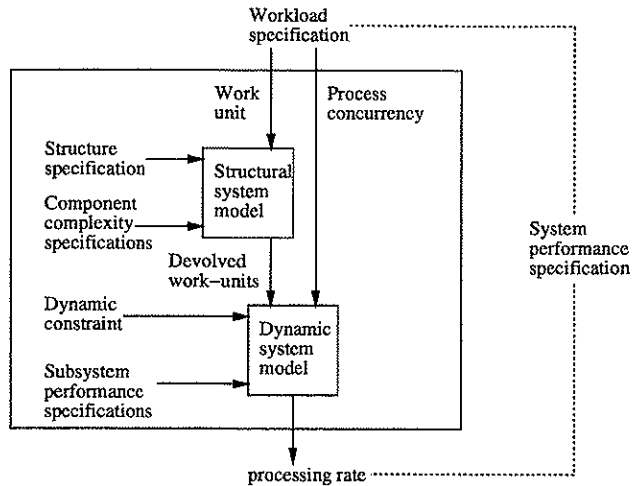


Figure 4.8: Hierarchical performance modelling [Hug95, p. 40].

The structure specification is considered in Section 4.6.1. System performance for one class is:

$$P_s = (\vec{u}, \mu_s(n), n_{s,max}) \text{ where } n = 0, 1, 2, \dots, n_{s,max}$$

Similarly, performance for a subsystem for one class is:

$$P_{ss} = (\vec{u}, \vec{\mu}_{ss}(n), n_{ss,max})$$

For the top-level system, $\vec{\mu}_s(n)$ is evaluated for a closed system level model and can be regarded as defining a FESC. The formulas must be applied *recursively* for all subsystems, until all the $\{P_{ss}\}$ for the corresponding \vec{u}_{ss} are available. See [Hug96, Hug95] for more information.

For multiclass workloads, the definitions above must be extended [Hug96]. Each class will be represented by a separate workload vector, and n must be replaced by a population vector \vec{n} describing the number of concurrently active customers of each class. For each value of the population vector, separate throughput rates must be defined for each class.

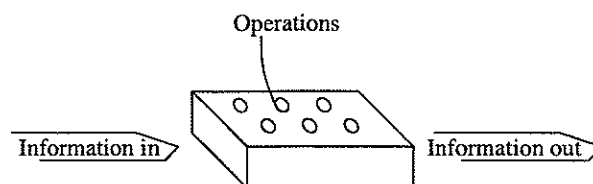


Figure 4.9: The virtual machine processes information with some operations.

4.4 Abstract Virtual Machine

The abstract virtual machine is a more exact name of the resources at several levels of abstraction which are modelled in SP. For a deeper understanding of SP, the concept of abstract virtual machine is important, but also hard to grasp conceptually. All the other concepts in SP are related to this concept. A virtual machine takes information in, processes this information based on some *operations* and sends the information out again as can be seen in Figure 4.9. Each operation may have parameters with data describing the operation further.

This machine is virtual because it is implemented in terms of software or human resources. The virtual machine concept was originally used in the design of operating systems [PS85]. An abstract machine makes no assumptions about the implementation, e.g. a Turing machine [Gil76]. It is illustrating to consider a real (as opposed to a virtual) machine which processes steel. Pieces of steel are taken into the machine and shaped according to some operations (i.e. buttons) on the machine. The final shape of the steel pieces is the result of a sequence of operations.

This sequence of operations in a virtual machine is the result of a *process* controlling the machine. A process is a partially ordered set of operations on some data. It is not a total ordering to allow for nondeterminism, which is necessary for parallel machines. Each operation of the virtual machine also starts a process, the so-called inner process, in contrast to the outer process which invoked the operation in the first place. This is illustrated in Figure 4.10. Suboperations are operations used by the inner process. A process can be a procedure in a programming language, a predicate in Prolog or a speech act between humans. In the steel example in Section 4.4, several buttons on submachines could in theory be pushed when one machine button is pressed.

4.5 Module and Component Specification

In SP, there are three concepts designed to specify levels of available information for an SP node, ranging from an ADT, when there is only superficial knowledge, to

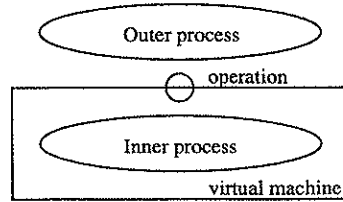


Figure 4.10: The outer process invokes an operation on the virtual machine and an inner process is started.

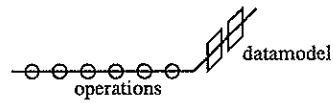


Figure 4.11: An abstract data type and a data model.

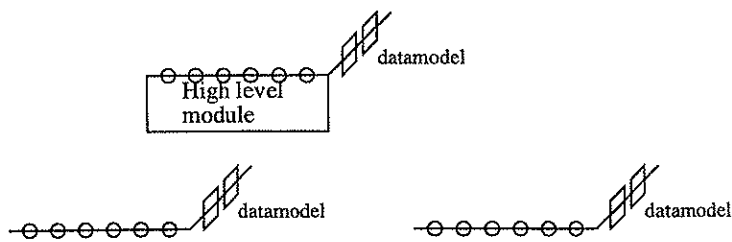


Figure 4.12: A module specification with data models.

modules and components where the knowledge level increases.

ADT A set of operations upon a common data structure is known as an abstract data type (ADT). This is completely abstract, i.e. it does not depend on some virtual machine. In Figure 4.11, an ADT is shown by a line with dots, where the dots symbolise operations. The boxes symbolise data structures. As an example, the ADT for the `Secretary_memory` in Figure 4.2 is:

$$o_{\text{Secretary_memory}} = \{ \text{get_internal_address}, \text{get_external_address}, \text{store_achieve}, \text{get_achieve} \}$$

This ADT has two data structures, namely *address-tuple* and *archive-tuple*. An ADT and data model taken together are termed an interface [MVH94].

Module A module is an ADT associated with a specific virtual machine, i.e., the lower-level ADTs on which it depends must be specified. This is illustrated in Figure 4.12.

Component A component is a module specification augmented with implementation dependent information about work complexity and compactness. While complexity specifications were introduced in Section 4.2.1.1, compactness specifications are described in Section 4.6.1.

4.6 Data Model

A data model determines the meaning of data and mutual relationships among the various items. In Figure 4.11, the data model is modelled as an orthogonal dimension to the operations. The data model is the interface to the memory of the virtual machine. If data should be stored beyond the current lifetime of the process, memory must be used. The memory may be viewed as an outer data structure implemented by a storage structure. The outer process accesses the outer data structure and the inner process accesses the storage structure. The storage structure implements the outer data structure. The storage structure is persistent just as the outer data structure. There is a relation between the data model in SP and PhM in PPP (PhM is described in Appendix B).

4.6.1 Compactness

Compactness specifications only apply for memory links. Compactness describes a relationship between a data structure and its storage structure in a lower level component [Min90]. The compactness specification G shows the amount of storage required for a single unit of data. The unit may be whatever is relevant at each level, provided consistency is maintained. $G_{\text{Disk}}^{\text{Lotus_Notes}}$ is the compactness specification for the relation between `Lotus_Notes` and the `Disk3` in Figure 11.4:

$$G_{Disk}^{Lotus_Notes} = \overset{blocks}{telex} [0.002 s + 1]$$

The above equation states that one document takes $0.002 s + 1$ blocks of storage allocation in the disk, where s is the size of an average document in characters. A document with 2000 characters will need $0.002 \cdot 2000 + 1 = 5$ blocks.

Each level of the memory chain has an extent limit specification [Min90]. The extent limit specification is a vector which specifies the maximum number of items of each storage structure in terms of its normal unit of allocation. For a 400 MB disk where the block is the storage structure and where the block size is 1 kB, the extent specification E_{Disk} will be:

$$E_{Disk} = \overset{blocks}{[400\,000]}$$

The devolved extent must be less than the extent limit:

$$E_{Lotus_Notes} \cdot G_{Disk}^{Lotus_Notes} \leq E_{Disk}$$

Compactness specifications and extent specifications are a relatively underdeveloped area in SP.

4.6.2 Workspace

The workspace is the temporary storage in a virtual machine during flow of information between operations of a process. The concept of workspace is orthogonal to the process concept. The outer process operates in the outer workspace just as the inner process operates in the inner workspace. Processes which get input from the outer workspace, other workspaces or the storage structure, operate on the inner workspace and place the results in the outer workspace, communicate with other workspaces or store data in the storage structure. As an example of a workspace in a computer system, the temporary storage of a procedure in the C programming language is the content of variables used during execution of this procedure. In Turing machines, the workspace is on the tape [Gil76].

The workspace of a person is his brain and in addition, temporary information on a blackboard, on a piece of paper or in a computer used by this person. A group of

people may also have a common workspace. In the Gas Sales Telex Administration Case Study in Chapter 10, paper is part of the workspace for the STATOIL organisation. Paper is processed by case workers and distributed by mail men, functioning as a way of communication between the internal processes inside of each case worker. For some operations, the paper will survive the lifetime of the operations, and can therefore be regarded as memory. This reflects the similar nature of workspace and memory. Persistence is the only difference between them.

4.7 Typing of Operations

It is especially the typing in SP which increases the possibility of spotting similarities between SP models, which again makes reuse simpler. As commented on in Section 5.4.5.2, reuse of performance models is important to save the cost of modelling. This section first describes the three basic operation types in SP, then some remarks are given on the fourth and seldomly used operation type in Section 4.7.2. A non-primitive module in SP is either non-distributed or distributed. The type rules for non-distributed SP models are explained in Section 4.7.3. Section 4.7.4 describes the rules for distributed SP models and also gives two examples of client-server SP models.

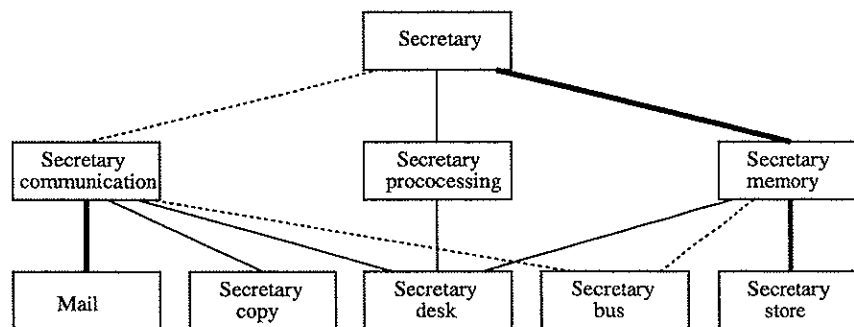


Figure 4.13: SP model for a secretary. Links between components are typed. Processing operations have solid lines, memory operations have bold lines and communication operations have dotted lines.

4.7.1 Basic Types

The basic types of operations in SP are processing operations, memory operations and communication operations. Typing of operations in SP is indicated by the thickness of the links in SP models as shown in Figure 4.13. Processing operations have solid lines, memory operations have bold lines and communication operations have dotted lines. These three types of operations conform to the main tasks of an

information system, which are manipulation, storage, and distribution of information [SK93]. The operation types are distinguished as follows:

Processing Processing operations transform information and operate in the local workspace of a virtual machine. This is in contrast to the memory and communication operations where information is either transported from or to the workspace, but not within the workspace.

Memory Memory operations or data access operations are used for access to and storing of data beyond the lifetime of the outer process. In Figure 4.13, `Secretary_memory` is the secretary as interface to the local archive which is a memory for the secretary.

Communication A communication operation transfers data between the outer and inner process when their workspaces do not interact. A communication operation is really a memory operation with low persistence and distributed access. This shows the blurring distinction between communication and memory, and may explain why it is easy to mix these operations in Lotus Notes.⁴

4.7.2 Discrimination

In addition to the three types described above, there is one additional type of operation, namely *discrimination*, which is internal to each process and therefore not visible in SP models. In practice, discrimination is lumped together with processing [Hug88]. Discrimination is one of the most subtle aspects of SP. Discrimination operations perform decisions concerning the process and is part of the so-called control process inside of the virtual machine [Hug88]. If an IF-statement in Figure 4.14 is executed in component A, this is discrimination in terms of the module A.

Only some part of the IF-statement in Figure 4.14 may be discrimination inside the CPU component. It may for example be necessary to *process* some data before a decision (discrimination) can be made. Thus, as we decompose, only some elements of discrimination at the upper level still continue to be so. One discrimination in the IF-statement in Figure 4.14 may also become several discriminations when decomposed, i.e. in addition to a test, some decisions regarding which memory segment that should be read may occur. If we decompose information processes down to the chip level, all information processing becomes discrimination, i.e. state changes on the chip.

⁴In Lotus Notes, described in Section 8.2.4 and Section 11.2, it is often easy to *send* information instead of just telling *where* in memory the information can be found. This gives higher workload on the computer system and therefore also degrades the performance of the humans using the computer system.

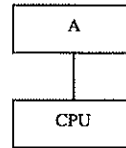


Figure 4.14: Discrimination in SP.

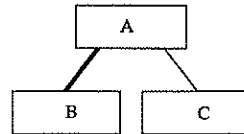


Figure 4.15: The non-distributed module A uses the submodules B for storage and the submodule C for processing.

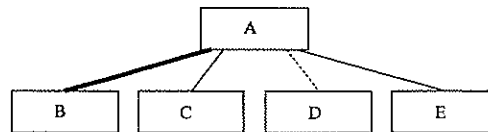


Figure 4.16: The module A is distributed.

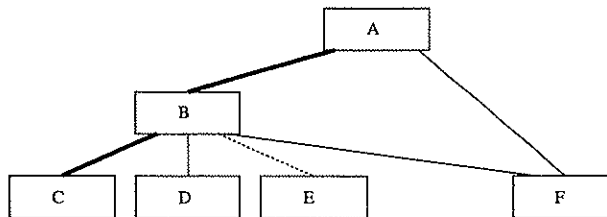


Figure 4.17: A client-server architecture.

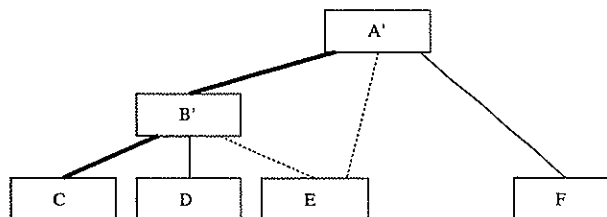


Figure 4.18: A variant of the client-server architecture.

4.7.3 Non-distributed Models

Non-distributed modules have one processing submodule and one memory submodule as shown in Figure 4.15 where operations in module A invoke operations in submodules B and C. Operations on submodule B are memory operations, and operations on module C are processing operations. This is indicated by bold lines for memory operations and by solid lines for processing operations.

4.7.4 Distributed Models

Distributed modules have (1) more than one processing submodule, (2) more than one memory module, or (3) both [Vet93, p. 93]. A simple example of a distributed module is depicted in Figure 4.16. Making SP models according to the principles for distribution is simple for models with only two levels: the module level and the submodule level. However, making SP models with more than two levels is hard, and will often take several iterations. Based on the basic relationships, non-distribution or distribution, more complex architectures can be made. In Figure 4.17, a client-server architecture is made by combining a non-distribution architecture with a distributed architecture.

Another representation of the client-server architecture is shown in Figure 4.18, where the intrinsic processing which was performed by component B in Figure 4.17, now is done by component A'. This representation is a transformation of the representation in Figure 4.17, where each level is non-distributed. The ' in A' signifies that the A component in Figure 4.17 is not similar to the A' component in Figure 4.18. This is expressed by the equation: $A' = A + (B - B')$, which states that A' performs more work than A. This work is equal to the work which was performed by B in Figure 4.17, and which is no longer performed by B' in Figure 4.18. The model of Lotus Notes in Figure 11.3 is an example of a more complex SP model. However, the extension from the client-server models above is straightforward.

4.8 Chapter Summary

SP is a method for modelling relations between resources at several levels of abstraction. The key feature of SP is the distinction between work and load. SP focuses on modelling of work which in contrast to workload is a static concept. SP may be combined with dynamic approaches, for example queueing networks, to deal with more complex performance problems. Links between SP modules may be a processing link, a communication link and a memory link. These links form an important part of making it more easy to discover similarities in different SP models, thus encouraging reuse.

A common criticism of SP is the high number of concepts. The reason for this complexity can be twofold:

1. SP introduces extra complexity which is not present in the domain from the beginning. In this case, SP only increases the workload on the analyst, without giving improved quality of the model in return.
2. SP captures complexity which is in the domain, but which has been hidden. It is now the choice of the analyst if he wants to use this complexity to improve the quality of the model in some respects.

My experience with SP is in line with the latter view. During SP modelling, the size of an SP model tends to increase when more and more of the complexity of the domain is understood. When sufficient understanding is reached, the complexity and thus the size of the SP model tends to decrease. Finished SP models are usually compact.

Chapter 5

Performance Engineering of Information Systems

For large projects, it is important to discover mismatch in performance *at an early stage* in the project, because counteracting performance problems late in the project is much more costly and time-consuming. This is similar to paying insurance to avoid high expenses later. With performance engineering of information systems, inadequate performance is discovered *before* integration and testing [Smi90] and could be defined as [Ale86]:

Performance engineering is the matching of software requirements to hardware options and defines computer capacity needs for given software requirements and operational conditions. Performance engineering ensures adequate computer capacity when the systems are completed and operational.

The first section outlines where performance requirements fit into the larger picture of non-functional requirements. Since performance engineering is part of the overall information system development, Section 5.2 describes information system lifecycle models. Then the motivation for performance engineering is discussed in Section 5.3. This section continues with an overview of the state of the art for performance engineering of information systems. The quantitative approaches which are mostly used, are described in Section 5.4, while one qualitative approach to performance engineering is described in Section 5.5.

5.1 Non-functional Requirements

All requirements which do not deal directly with the functional relation between the environment and the system, are defined as non-functional requirements. Broadly

speaking, for non-functional requirements there is no direct route from specification to implementation [Fin91]. Thus, non-functional requirements are in general hard to retrofit, i.e. to fit in after the product has been finished [DBC88]. The distinction between functional and non-functional requirements is not clearcut, however. As a design area becomes more mature, some of the non-functional requirements will become functional, e.g. the non-functional requirement of user friendliness has to some extent been taken care of by advanced user interfaces specified as part of the functional requirements.

There is no generally agreed upon taxonomy of non-functional requirements [Chu93], but non-functional requirements will at least include effectiveness requirements, interface requirements and adaptability requirements.¹ Some authors make a distinction between customer-oriented (observable by the customer) and system-oriented (only internal to the computerised information system) non-functional requirements [Chu93, KKP90] or between product-oriented (oriented towards the final product) and process-oriented (focus on the development process) non-functional requirements [Chu93]. In the ideal case, the tradeoff between non-functional requirements is made explicitly and not left to intuition. Such an explicit tradeoff process is illustrated in [SW93] and [MCN92] from a quantitative and a qualitative modelling point of view, respectively.

5.2 Lifecycle Models for Development of Information Systems

Performance engineering will be embedded in an information system development lifecycle. This section presents some well-known and also some promising lifecycle models and comments on how performance engineering fits into each lifecycle model.

5.2.1 The Waterfall Model

The waterfall model in Figure 5.1 forms the basis for most lifecycle models used in practice today [Boe88]. The advantage of the waterfall model over the conventional code-and-fix approach is to encourage design before coding and to increase the quality of the documentation, which in particular will make maintenance easier. With clearly separated phases where documentation is the end product of the previous phases, project control is also easier than with the code-and-fix approach. Each phase culminates by verification and validation to eliminate as many problems as possible in the lifecycle products. Iterations of earlier phases are to the extent possible performed in the next succeeding phase, which is illustrated by the arrows in Figure 5.1.

¹Adaptability requirements determine how easy the system is to adapt to changes in the environment: expandability, flexibility, interoperability, portability and reusability [Chu93, KKP90].

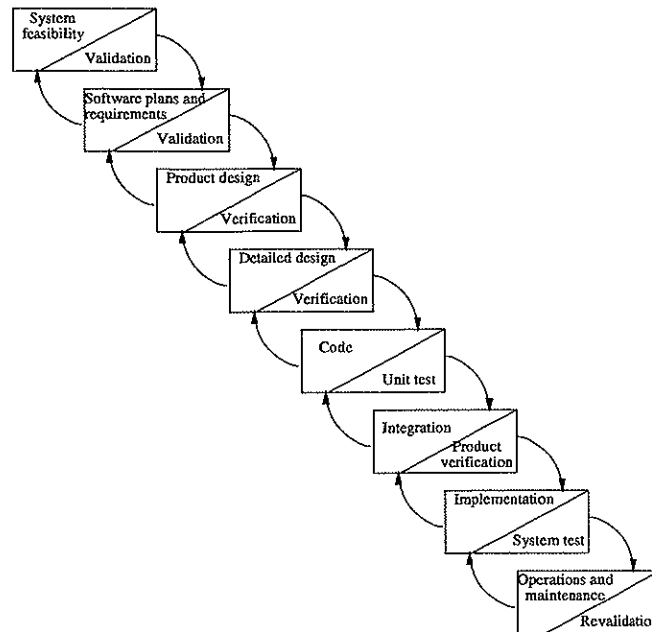


Figure 5.1: The waterfall model of the software lifecycle [Boe81].

Below, each phase is considered in sequence [Boe81]:

System feasibility Defining a preferred concept for the software product, and determining its life-cycle feasibility and superiority to alternative concepts.

Software plans and requirements A complete, validated specification of the required functions, interfaces, and performance for the software product.

Product design A complete, verified specification of the overall hardware-software architecture, control structure, and data structure for the product, along with such other necessary components as draft user's manuals and test plans.

Detailed Design A complete, verified specification of the control structure, data structure, interface relations, sizing, key algorithms, and assumptions of each program component.

Coding A complete verified set of program components.

Integration A properly functioning operational software product composed of the software components.

Implementation A properly functioning operational hardware-software system, including such objectives such as program and data conversion, installation and training.

Operations and Maintenance The system runs in production, and is maintained.

Performance engineering in the waterfall model is elaborated in more detail in Section 5.4.1, where the SPE approach of Connie Smith is outlined. The basic problem with the waterfall model is the emphasis on fully elaborated documents as completion criteria for early defined requirements and design phases, which in turn may lead to elaborate specifications of poorly understood requirements [Boe88].

The waterfall lifecycle model has been extended to the structured lifecycle model, where the degree of formality is increased by the use of structured analysis and design assisted by semi-formal conceptual modelling like data flow diagrams, entity relationship modelling and state transition diagrams.

5.2.2 Extending the Waterfall Model with Prototyping

As an extension of the waterfall lifecycle model, prototyping may be used. Functional prototyping is typically used in information systems engineering to reduce uncertainty, e.g. to get a better understanding of requirements. The development process using functional prototypes are usually highly iterative with much user participation. The two basic forms of functional prototyping are evolutionary prototyping or throwaway prototyping [Von90]. The evolutionary prototype eventually evolves into a full information systems, and focuses initially on well-understood requirements [DBC88]. In contrast, throwaway prototyping (or rapid prototyping) focuses on the poorly understood requirements and are thrown away when requirements are stable. The user interface may for example be investigated with a throwaway prototype. For a taxonomy of prototyping, see [Lin93].

In contrast to a functional prototype, a performance prototype will try to emulate the hardware and the software of the target system, and is therefore complementary to the functional prototypes as depicted in Figure 5.2. A performance prototype may be generated from detailed design specifications [Hug84]. A prototype interactor which collects statistics during a user dialog may later emulate the workload. Compared with the method advocated in this thesis, this method is quite time-consuming, and also requires the target platform to be physically available. However, for non-standard or untried parts of the design where no validated platform models are available, (throwaway) performance prototyping may be the only alternative.

5.2.3 Extending the Waterfall Model with Incremental Development

Incremental development is a possible extension of the waterfall lifecycle model. Incremental development is the process of constructing a partial implementation of the total system, and then slowly adding increased functionality or performance. The term evolutionary development is also used. In contrast to evolutionary prototyping, incremental development presupposes that most of the requirements are understood before the process starts. The two methods may of course be combined. Like

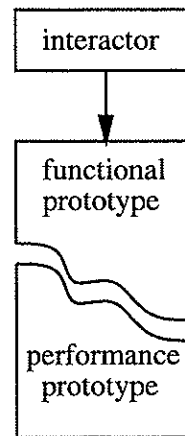


Figure 5.2: The complementary functional and performance prototypes may be driven by a prototype interactor.

the code-and-fix approach, incremental development may lead to hard-to-change spaghetti-code. This is especially a problem when the projected system happens to be integrated with an existing system [Boe88]. Performance problems may be hard to discover at an early stage during incremental development.

5.2.4 Operational and Transformational Development

The basis of operational development is a formal executable language, enabling validation [Zav84]. One problem with the operational model is inadequate performance, which is expected since one of the objectives of the operational method is to delay binding of resources as far as possible. The idea is to increase the performance of the operational model through transformations. These transformations preserve the external behaviour of the executable model, and are also written in the same modelling language. However, more details are included, making the model better suited for implementation. There is not much practical experience with the operational development method, but this method is promising since hierarchic or top-down development may be possible. This is in contrast to the incremental development approach which advocates bottom-up development.

The operational method is quite similar to the transformational method, with the exception that by using the operational method, automated support for the transformation is more stressed than with the transformational method. Both the operational and the transformational approach use prototyping actively. Whereas the prototyping techniques mentioned in Section 5.2.2 use different languages for specification and prototype development, this is not necessary in the operational approach since the specification itself is executable.

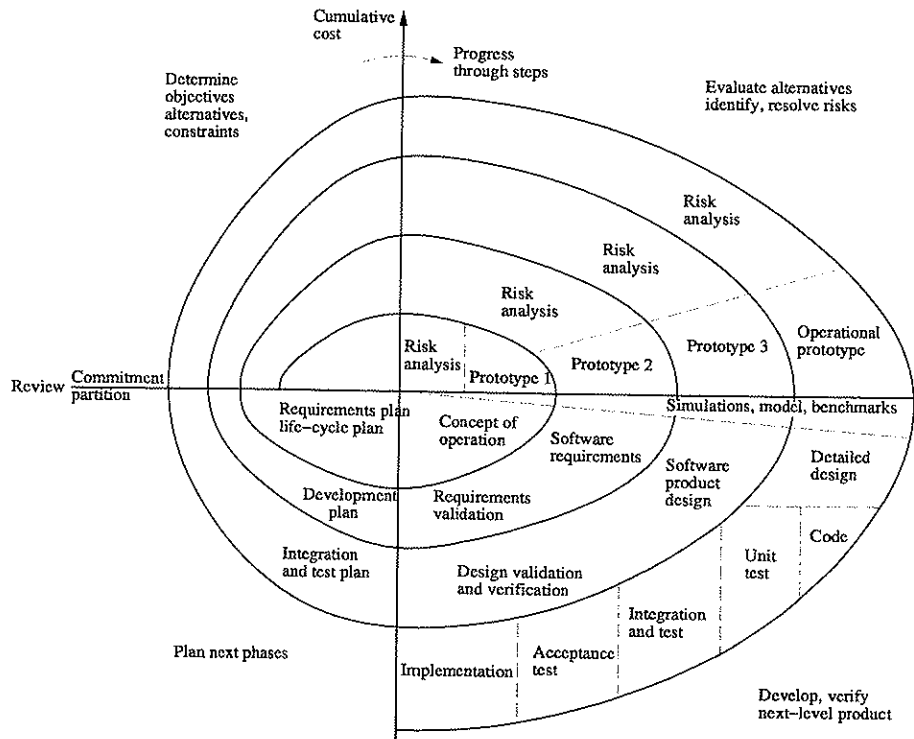


Figure 5.3: The spiral model [Boe88].

5.2.5 The Spiral Model

The basis of the spiral development model is analysis of risk. Development proceeds through the spiral as illustrated in Figure 5.3. Each iteration of the spiral starts by identifying objectives for the product being developed (e.g. the performance), alternatives (for implementing this product) and constraints (e.g. cost, schedule, non-functional requirements). The alternatives are then evaluated relative to the objectives and constraints. The risks found during this evaluation is resolved with prototyping, simulation, analytic modelling, benchmarking etc.

Development proceeds in a spiral fashion where effort is directed to high-risk areas during each turn of the spiral, for example through extensive prototyping. Low-risk areas may even be neglected. Depending on the risk structure of the project, development can reduce to most other methods like waterfall, both evolutionary and throw-away prototyping, or operational/transformation. The focus on the risky part of the system is a strong point because this gives early attention to these parts, without waste of resources and time. But the basic problem with this approach is that risk identification and resolution require experience. Thus, the method is hard to use by inexperienced personnel. Moreover, the spiral model may foster the development of specifications which are not necessarily uniform.

The hierarchical spiral model is an extension of the spiral model [Iiv90a, Iiv90b]. In the hierarchical spiral model, the emphasis on conceptual modelling is stronger than in the spiral model, and three levels of modelling is described: (1) organisational level, (2) conceptual level, and (3) technical level (this is fairly similar to the three levels in OSSAD in Section 8.1.3).

5.2.6 PPP Method

The experimental CASE tool PPP forms the language, method and tool environment of this thesis. Whereas the PPP languages and CASE environment are outlined in Appendix B, the PPP method is explained here. The development of a computerised information system is perceived as a mapping between the manual system and the computerised system. A model of the manual system precedes the design of the computerised information system as shown in Figure 5.4.

The PPP method is top-down where models are developed in an incremental and iterative manner. The iterative and prototyping development discipline which is used in PPP is similar to the spiral development model. During the early phases of information system development, the focus is on understanding the problem domain and not so much on making requirements for the finished product [LSS94]. Later, analysis and design are highly integrated. Stating which parts of the manual system should be automated determines the automation border. The parts to be automated are then decomposed and specified to a sufficient level of formality for code to be generated automatically.

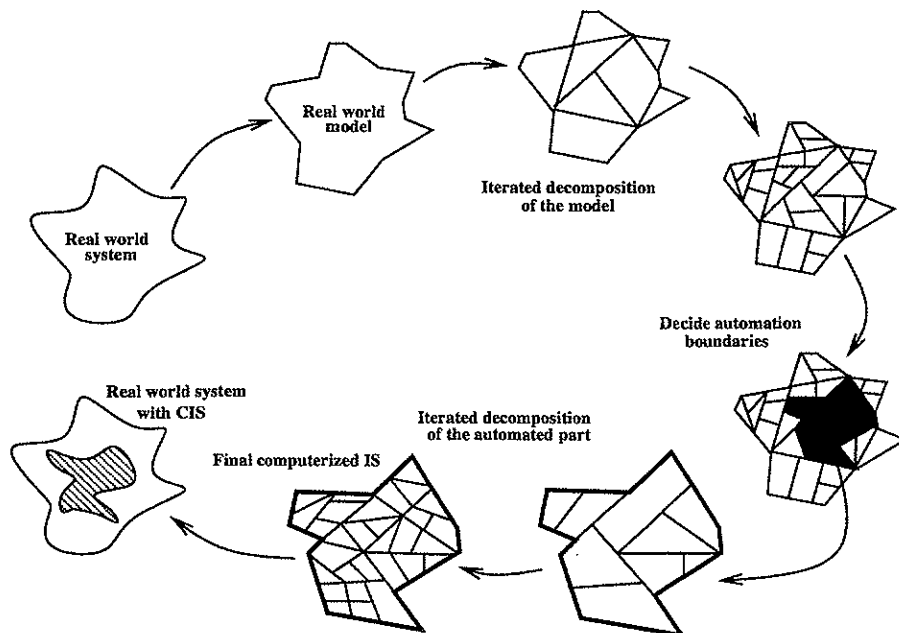


Figure 5.4: The PPP method [GLW91].

Due to the tightly coupled languages, models in one language may generate initial models in another language, e.g. a process model may generate an initial version of a data model. (See Section B.1.2 about the process language in PPP, and Section B.1.2 for the data modelling language in PPP.) Prototyping [Lin93], model execution [Wil93], model explanation [Gul93], model verification [Yan93], configuration management [And94], and support for several views [Sel94] are integrated parts of the PPP method. See also [Kro95, p. 152] for more information on the PPP method. As an experimental method, the PPP method is not sufficiently tested in industrial environments.

5.3 Motivation for Performance Engineering

Even if hardware costs decrease, performance will not disappear as a design problem, which is illustrated by the saying: "Software gets slower faster than hardware fast." The demand for software and hardware has until now increased at a larger rate than the reduction in hardware costs [Opd92]. The primary motivation for performance engineering of information systems is to decrease the cost and risk of system development. When performance engineering successfully augments information system development, timely completed information systems which satisfy performance requirements can be realised.

Tuning is often appropriate when performance problems can be localised to specific areas of a system, e.g. denormalisation of database tables, adjusting system parameters, limiting number of file openings and closing, etc. [Fox89]. However, this so-called “fix-it-later approach” on the implemented system will not save the design from inadequate design decisions [Smi90] or overconstrained interpretations of performance requirements [Fox89]. It is easy to see the similarity between the fix-it-later approach towards performance and the code-and-fix approach to overall software development: both approaches give developers full freedom but little help.

An important characteristic of performance engineering is to facilitate communication between phases in the lifecycle. Information which is needed early in the lifecycle are transported from later phases of preceding projects. As stated in the preface, *overview* is crucial for this communication to succeed. With an overall method for performance engineering during development of information systems, only critical parts of an information system need to be designed with performance in mind. For the other parts, no extra effort is needed. Thus, performance engineering may reduce the cost and risk of designing an information system with acceptable performance, since performance optimisation is only applied for the critical parts of the system. Moreover, this optimisation could be done *earlier* with the overview which performance engineering offers. For more discussion of cost/benefit analysis of performance engineering, see Section 5.3.1.

Despite the benefits, performance engineering is still either consciously or unconsciously neglected because [Bel87, BF87b]: (1) Functional problems receive more attention. (2) Resistance towards using a method; the aspect of art disappears. (3) Lack of skills, because there are few developers with performance engineering skills from similar projects. (4) Hard to see benefit in advance. Like insurance, performance engineering only pays off when accidents happens. (5) High cost, e.g. interviews take time for busy people. (6) Resistance towards redesign: early performance estimates may reveal need for redesign which may be hard to sell to management because of the cost impact. (7) Performance is somebody else’s responsibility. In addition, an external performance engineer will often have problems getting workload information, which may be business confidential.

In general, however, there is a definite trend in industry towards “turn-key” or total project engineering. Customer and market requirements to ensure the performance of the total delivery are driving forces. On this basis, it is believed that the motivation for performance engineering is going to increase.

5.3.1 Tradeoffs in Performance Engineering

When doing performance engineering, the most important factors in this tradeoff are, cost, accuracy and benefits as described in Section 5.4.5.2. Both performance modelling and measurements have cost implications. For performance modelling, the costs are roughly:

Operational The cost of not using the measured computer system for other purposes. For some measurements, an idle machine is needed. Therefore, it will be necessary to use some sort of test system.

Manpower The time of qualified performing needed to perform measurements modelling will of course have a cost. In addition, valuable manpower resources are tied up.

Equipment Extra software (e.g. measurement programs like SPM/2) and extra equipment (e.g. measurement hardware to measure network traffic) may have to be purchased. Often, however, this kind of equipment will be needed anyway even if fix-it-later is used.

In practice, the largest cost factor is likely to be the cost of qualified manpower. Costs of performance modelling will consist of:

Delay in project Performance modelling takes time, and may slow down the project.

Manpower People with performance modelling competence may be expensive to hire.

Equipment Tools for performance modelling are not off-the-shelf software and may therefore be expensive and hard to get.

As for performance measurements, manpower costs are likely to be highest also here.

Accuracy is important in each submodel and especially for the overall performance model. Overall performance model accuracy can only be assessed accurately after implementation, during verification. The whole performance model must be balanced in terms of accuracy. During the early phases of development, there are several reasons for using simple performance models, e.g. a simple model is easier to make and to modify while a complicated model will often be delayed relative to development [BF87b]. Operations at a high level of abstraction are generally more vague and also harder to annotate in terms of resource demands, because the link to implementation is looser than at a lower level of abstraction. However, since the number of low-level processes is higher, this requires more manual work. Thus, there is a tradeoff between cost of performance engineering and accuracy.

The benefits of performance engineering are hard to estimate, because this depends on the cost of shortcomings in a system with performance problems, which is hard to estimate, because it is hard to carry out exact experiments here. The cost of performance problems will most likely not be put on print, e.g. this project costed 20 % more because of performance problems. Besides, performance problems will often result because of other problems. In a controlled project, where performance engineering is performed, performance will be under control, and there will be few performance problems. Often, performance engineering will not have large effects the first time it is applied. But increased awareness of performance issues and compilation of statistics will often have long-term effects.

The most dominant benefit of using performance engineering is reduction in the need for manpower to do fix it later. Slowing down a project, as a result of fix it later efforts, will not affect only the person performing the programming, but also the rest of the project members and in the end, the rest of the organisation. Time will often have a value also independent of manpower cost: if the time to do fix-it-later is reduced, this will open up new possibilities, irrespective of the cost involved. Time to market is a critical factor in industry. Reduction of development time increases the market potential for a given product, which is overall beneficial.

The primary success criteria for performance engineering is: $cost < benefits$. Cost and benefits will be a function of accuracy. If accuracy is below a certain limit, the performance models and measurement cannot be used for anything useful and there will be no benefits, even if there is a cost involved. Cost/benefit for performance engineering of information systems are also considered in [Smi90]. Cost/benefit calculations for tuning are described in detail in [FSZ83] and may also add more details to the discussion above.

5.4 Quantitative Performance Modelling

In quantitative performance modelling, numbers are used as the basis for performance engineering. The focus of quantitative performance engineering is not so much on accurate performance prediction, but more on removal of unrecoverable bottlenecks [Fox89]. Performance engineering activities were commented on in the context of the lifecycle models earlier in this chapter. Below, the most promising approaches to performance engineering are considered. The SPE approach has an elaborated method and this aspect is therefore given attention, while for the HIT approach, the interesting aggregation philosophy is outlined.

5.4.1 SPE

Connie Umland Smith was one of the first to combine analytic models of software with analytic models of hardware [SB80, Smi86]. Her Software Performance Engineering (SPE) approach uses software execution models to model software, and system execution models to model hardware as depicted in Figure 5.5.

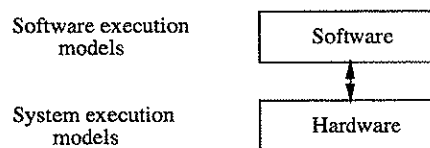


Figure 5.5: Software execution models are used to model software, while system execution models model hardware.

The software execution models are extensions of program flowcharts and the system execution models are extensions of queueing networks [Smi90]. The SPE method is shown in Figure 5.6, using the *software execution language of SPE*. The SPE method in Section 5.2 augments existing lifecycle models rather than replacing them. Below is a brief explanation of the method:

Define SPE assessments for lifecycle phase Define performance objectives for the current lifecycle phase.

Create concept for lifecycle product Make models appropriate for the lifecycle phase, using the principles for creating response software explained in Section 5.4.1.1.

Gather data By measurements of existing applications and by performance walkthroughs where users representatives, software representatives and performance analysts meet to discuss the information to put into the models.

Construct & evaluate appropriate model Construct and evaluate software execution models and system execution models.

Report results During performance walkthroughs.

Alternatives preferable If the models indicate that performance goals will not be met, alternatives are found and cost-estimated. If feasible and cost-effective alternatives are found, the concept for this lifecycle is changed according to the principles for creating response software in Section 5.4.1.1. The performance goals are revised if no feasible and cost-effective alternatives are found.

Complete lifecycle product The lifecycle product is revised according to the findings in the previous steps.

Verification & Validation An important part of the SPE method is continued verification of the model specifications and validation of performance model predictions. Performance models are replaced by prototypes and implementations at an early stage for critical parts of the software.

Enter next phase The steps above are repeated for each phase in the lifecycle.

Integration with conceptual modelling is not an integrated part of SPE. The SPE method has been used on such systems as client/server applications [SW94], real-time systems [SW93] and MIMD computers [SL82].

5.4.1.1 Principles for Creating Responsive Software

Smith presents seven principles for making software with good performance [Smi90]. Most designers would more or less intuitively use these principles. Making the principles more explicit is likely to increase the practical use. As a consequence of

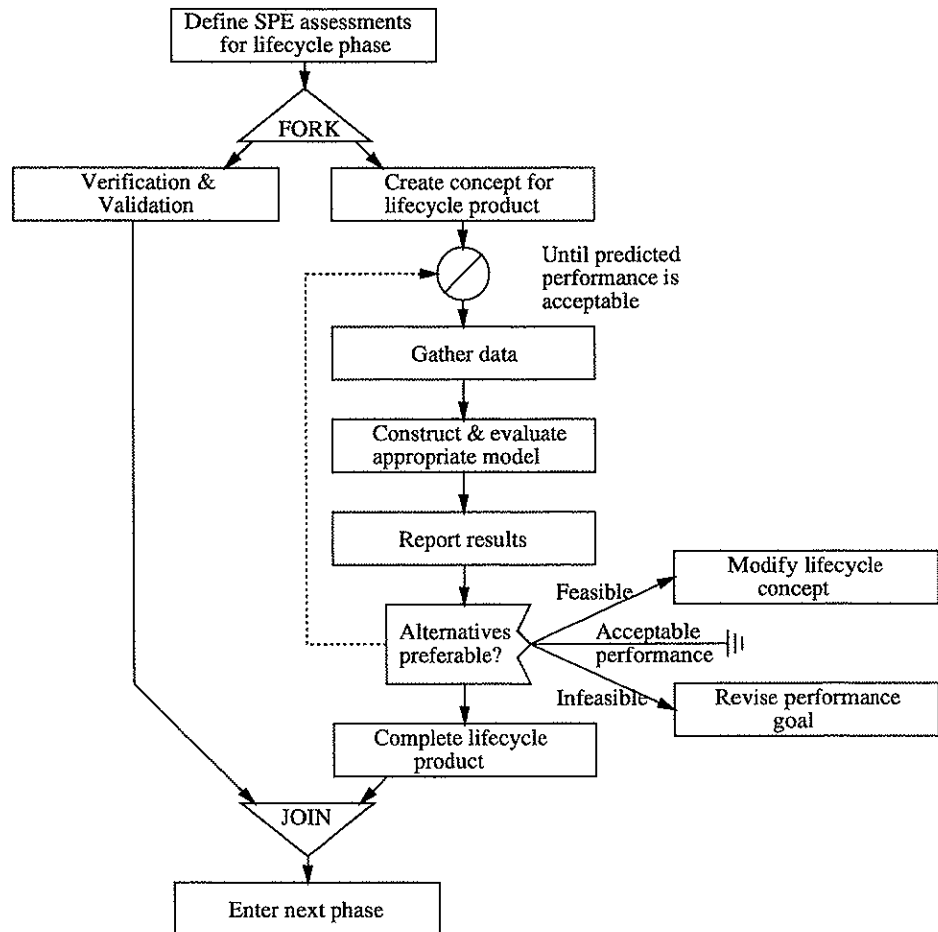


Figure 5.6: The SPE method [Smi90].

our rigid definitions in Chapter 4, these principles are easy to explain. Below it is indicated if the principles could be applied for only one process at a time or if they depend on synergy between several processes:

Fixing-point (independent) Execute processes as early as found cost-effective.

The time of execution is termed the fixing-point, because at this point in time the connection between the process and the result is fixed.

Locality-design (independent and synergy) Use resources which are close to the process in terms of time, space, degree and effect.

Processing versus frequency tradeoff (independent) Minimise the product of work and load for each process.

Shared-resource (synergy) Minimise the sum of administration and holding time for shared resources.

Parallel processing (synergy) Minimise the sum of administration time and execution time for parallel processes.

Centring (independent) Identify dominant processes and reduce their work.

Instrumenting (-) Monitor performance through instrumenting processes and resources.

The principles which depend on synergy will require information about performance and not only about work. The principles could be applied at several levels of abstraction and also at several points during the lifecycle.

5.4.2 LQM

Layered Queueing Models (LQMs) are designed to provide performance estimates for distributed systems [RS95]. With LQMs, it is possible to study the effects of changes in requests between processes, the number of instances of processes, the internal level of concurrency within processes, and the placement of processes on processors. LQMs have separate models for software contention and for (hardware) device contention, and can model systems with contention both for software and hardware resources.

The graphical notation for software processes and (hardware) devices are shown in Figure 5.7. In this figure, each parallelogram is a group of one or more processes. A group of processes are processes which are statistically identical. Each circle in the figure is a (hardware) device. Directed arcs indicate requests for service from a calling group to a serving group or device.

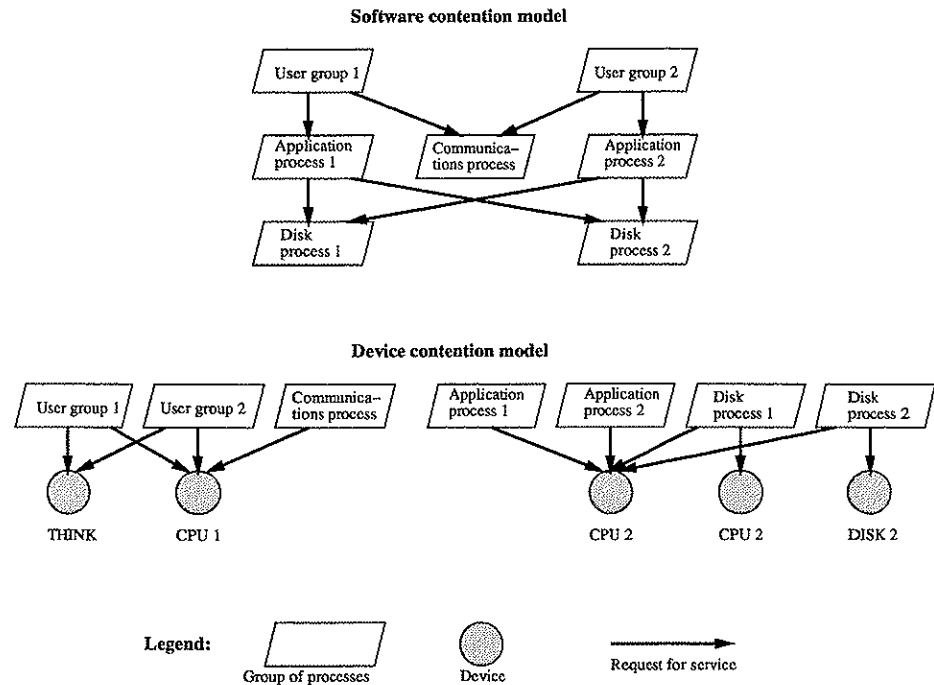


Figure 5.7: Example of a software process architecture [RS95].

LQMs can be solved with MVA based algorithms. For LQMs, in addition to the standard parameters for queueing network models, the average number of visits a process makes to other processes and process scheduling disciplines must be specified. Two solution methods for LQMs are described. The Method of Layers (MOL) is described in [RS95], while The Stochastic Rendezvous Network (SRVN) is outlined in [FHM⁺95]. SRVN is quite similar to LQM, but employs a simpler Bard-Schweitzer MVA approximation, instead of the Linearizer algorithm used in MOL [RS95].

Compared to SP extended with queueing networks, LQM focuses more on dynamic modelling. Whereas modelling of contention both in software and hardware is an area for further research in PPP/SP, it is more developed in LQM. On the other hand, static modelling is not so much in focus in LQM. An integration of SP and LQM may be possible. SPs graphical diagrams could supplement the layered queue models.

My colleague and SP-forerunner Vidar Vetland is a co-author of at least two LQM papers about distributed application development. Parameter capture is discussed in [RV95], and scalability is an issue in [RVH95].

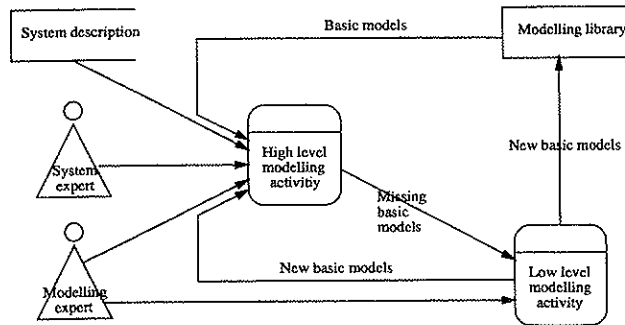


Figure 5.8: The overall COMPLEMENT process model (Adapted from [CVT+92]).

5.4.3 COMPLEMENT

The COMPLEMENT² project describes performance engineering of real-time and embedded systems [PBA+93]. As part of the COMPLEMENT project, Ayache et al. explain annotations of HOOD design objects with performance information [AC92]. Performance requirements are included in HOOD objects, which together with annotations of estimated resource consumption and execution characteristics, make it possible to design high-level performance models which are meaningful for designers. Static performance modelling is not included in the final prototype, even if some thoughts along these lines have been done in internal reports.

A process model for making dynamic models as an interaction between designers and modelling experts is described [CVT+92]. This process model focuses on the interaction between the designers and modelling experts as shown in Figure 5.8. Modelling experts construct low-level models with standard modelling techniques like Petri-nets and queueing networks. Generic low-level models make it easier to create specific instances of low-level models, termed basic models. A modelling library assists the modelling expert in creating new generic and basic models. These high-level models are elaborated enough so that meaningful performance models could be made. The designers compose a high-level model of the system using basic models from this library. This high-level model contains all the necessary information for generation of the low-level model and is an intermediate representation between the user and the low-level model. The low-level model is a representation of the high-level model, by using a modelling technique. When basic models are used to model a system, and when this system is implemented, measurements become available. These measures are used to calibrate the basic model and indirectly also the generic models. This activity is an important part of the interaction between the designers and the modelling experts. Three case studies are mentioned, and the result of these case studies should refine the approach [CVT+92].

²Partially funded by the Commission of the European Communities as ESPRIT II project number 5409.

5.4.4 HIT

The modelling tool HIT (Hierarchical Evaluation Tool) [BF87a, BMW88, Bei89] is developed at the University of Dortmund by Beilner et al. In HIT, different submodels can be solved using different solution techniques, ranging from simulation to exact analysis. The submodels are combined using aggregation techniques. Thus, the focus in HIT is on several solution techniques, so-called heterogeneous modelling. The language of HIT is both graphical and textual. The graphical models in HIT resemble SP models, but they miss typing in processing, memory and communication types of operations. Since there is no explicit notion of work in the HIT, there is no explicit static modelling in HIT either. HIT is a commercial tool.

5.4.5 Extensions of SP

In the Information Systems Group at NTNU, two theses have contributed to a deeper understanding of the SP method, and are outlined below.

5.4.5.1 Opdahl's Framework

Andreas Lothe Opdahl has made a framework for extending common CASE-tool with performance parameters during design, enabling performance engineering integrated with the normal design process of computerised information systems [OS92, OVBS92, Opd92]. The practical feasibility of the approach was demonstrated by implementing the ideas in the experimental CASE-tool PPP, and by using this tool on the Blood Bank Case Study described in Chapter 7. As described in the Blood Bank Case Study and in Figure 5.9, process models must be annotated with three types of parameters to enable performance engineering:

Workload intensities For all the operations which are invoked by the organisation surrounding the computerised information system, workload intensities must be specified. See Section 7.1.4 for a practical example.

Branching probabilities Describing the average probability of triggering other processes. See Section 7.2.1.1 for a practical example.

Resource demands Every primary process must be annotated with resource demands in terms of a platform. This platform will typically be an SP ADT.³ See Section 7.2.1.3 for a practical example.

Based on these three types of performance parameters, together with a platform model, performance of the computerised information system can be predicted as

³See Section 4.5 for an example of an abstract data type (ADT) in SP.

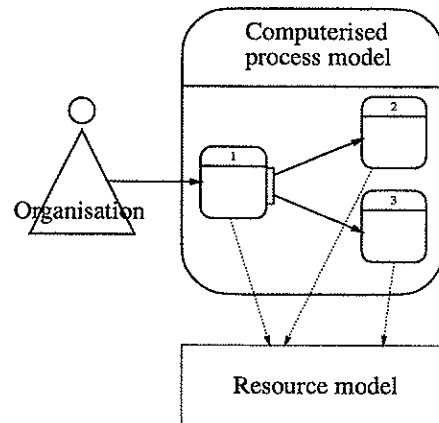


Figure 5.9: The process model must be annotated with workload intensities, branching probabilities and resource demands.

illustrated in the Blood Bank Case Study. Parameters for aggregated processes may also be used if enough parameters are not available for decomposed processes.

Sensitivity analysis to find the most critical parameters receives much attention in Opdahl's thesis. The idea behind this sensitivity analysis is primarily to use approximate parameters in a first run of the model. The sensitivity analysis will show the *critical* parameters, and more efforts could therefore be used to estimate them in *successive* runs of the model. This enables system developers to assist in estimation of performance engineering parameters even if their understanding of performance engineering is low.

5.4.5.2 Vetland's Framework

Vidar Vetland elaborated on the SP framework for composite modelling of work in software and hardware both by deepening the theoretical understanding of SP and giving real examples [Vet93, VHS93a, VHS93b]. When the software and hardware are divided in several models, parameter capture, accuracy, cost and reusability can be improved. In his thesis, Vetland describes these tradeoffs [Vet93, pp. 143 – 161]:

Feasibility of measurement Limitations in measurements can be classified as follows: (1) The inherent problems, (2) the practical problems, and (3) the availability of information about structure and performance. The inherent problems of workload modelling (operational variety, data dependence, and sequence dependence) [Hug96] manifest themselves at each layer of composite workload models. The practical problems can usually be solved by extensive tool support for measurement and modelling. The availability of information is crucial since tools are of no help if the required information itself is not available.

Compatibility with contention model All operations on hardware and software resources with contention must be modelled explicitly in the work model, thereby increasing accuracy and cost. Vetland elaborates on the interaction between SP and contention modelling using examples.

Reusability will decrease cost, but this reduction depends on the extra measurement experiments and changes of other work model components that have to be undertaken when it is reused [OSV93]. If work depends severely of the state parameters (see Section 4.2.1.2), the cost of reuse must also include the cost of modelling these parameters. The amount of extra measurements needed during reuse will determine both accuracy and cost of reuse.

Risk of error propagation Will increase when an additional level of work model components is introduced. This risk of error propagation is related to the worst case accuracy, so if additional levels are introduced, the complexity functions for each level must be more accurate. On the other hand, introducing more operations does not affect the risk of error propagation.

Measurement cost Consists of the number of measurement runs which must be prepared and performed, and the number of complexity functions needed. When one more level of work model is introduced, more complexity specifications and complexity functions are needed.

In a recent paper, Vetland et al. has made a complexity specification for a TCP/IP implementation [PVR95].

5.4.6 Taxonomy of Quantitative Approaches

A taxonomy of the languages for performance engineering is shown in Figure 5.1. In this taxonomy, the SP method is combined with the PPP CASE tool and queueing networks as described in Section 5.4.5.1. This taxonomy is slightly biased, since it focuses on the strong features of the PPP/SP combination.

When comparing the approaches, one should separate the conceptual and the implementation issues. Even though the PPP/SP approach [BOVS92] and the SP approach [Vet93, PVR95] have some case studies, the SPE approach is for example more mature than the PPP/SP approach. HIT is also more mature in this respect, while LQM and COMPLEMENT seems fairly equal to PPP/SP in terms of practical experience.

Approach	SPE	LQM	HIT	COMPLEMENT	PPP/SP
Interaction with design tools	indirect	direct	indirect	direct	direct
Typing of component interfaces	no	partial	no	no	yes
Hierarchical approach	implicit	explicit	explicit	implicit	explicit
Static modelling	partial	partial	partial	partial	complete
Dynamic decomposition	open	MOL/SRVN	aggregation	open	open

Table 5.1: Taxonomy of approaches to performance engineering.

Interaction with design tools Information systems are often developed with design or CASE tools. An important issue is if the information system models used in CASE tools are easily integrated with the performance models in question. COMPLEMENT and PPP/SP have a strong coupling to CASE tools which is indicated in the scoring. LQM is also designed with easy coupling to design tools in mind. SPE and HIT do not seem to have this strong coupling. They seem to be more oriented towards the performance analyst, compared to PPP/SP and COMPLEMENT, which focus more on the developer of information systems.

Typing of component interfaces SP is the only approach with explicit typing of operations (c.f. Section 4.7). This feature gives rules for model construction, and increases the possibility of finding similarities with other (older) models, making reuse more likely. Typing rules also give information about missing parts of the model and therefore function as consistency checks. Interaction with design tools may therefore be more formal. LQM separates software modelling from hardware modelling, which is a step towards explicit typing. In the other approaches there is no typing.

Hierarchical approach The hierarchies in SP and SPE are complementary. In SPE the hierarchies are implicit.⁴ The implicit view of hierarchies in Software Execution Models is intuitive for simple software, but for larger designs the explicit hierarchic view of LQM, SP and HIT compresses the details and increases the overview. See Section 6.2 and 6.5 for an elaboration.

Static modelling Static modelling builds on a separation between work and load. This feature is complete in SP, but only partial in the other approaches. SP also has the static concepts of compactness and extent (c.f. Section 4.6.1).

Dynamic decomposition All the approaches rely on dynamic modelling, using queueing networks or simulation. LQM was specifically designed to model both contention in software and hardware simultaneously, with the Method of Layers (MOL) or Stochastic Rendezvous Network (SRVN). HIT uses aggregation of separable subsystems. For the other approaches, the choice of decomposition method is open, but historically only aggregation seems to have been used.

Other complementary pairs of models could be envisaged. Thus, possible candidates will therefore also be SPE + SP, HIT + PPP or LQM + PPP/SP. This has not been considered in more detail.

⁴Implicit hierarchies are also normal in CASE tools, e.g. PrM in PPP. Based on PrM diagrams which are decomposed, explicit process hierarchies can be made (c.f. Section B.1.2).

5.5 Qualitative Performance Modelling

All the approaches described until now have been quantitative, i.e. focusing on using numbers to facilitate in performance engineering. Quantitative performance engineering is also dominant in practice and in the literature. Nixon has presented a qualitative approach [Nix94b, Nix93, Nix91] which is part of the comprehensive TELOS [MBJK90] and DAIDA [JMSV92] framework for development of information systems which also deals with other non-functional requirements [MCN92, Chu91, CKM⁺91]. Performance requirements are represented together with other non-functional requirements as interrelated goals [MCN92]. During the implementation process, performance goals, decisions, and their effect on goals are organised into goal graphs, helping the developer to select among alternatives and to justify implementation decisions, e.g. related to database definition and development [Hys91]. Principles from [Smi90], outlined in Section 5.4.1.1, are used in this decision process. Developers are aided by having a catalogue of decomposition methods which allow them to select aspects to explore and focus upon.

5.5.1 Comparing the Quantitative and Qualitative Approach

The qualitative approach does not depend on getting numbers which are problematic in the quantitative approach. On the other hand, numbers are needed to get overview, e.g. which parts of the system are important and which are not? It is also more easy to make decisions based on quantitative information than on qualitative information, especially for large systems. Thus, the quantitative and qualitative approaches are complementary, each contributing to the other. Therefore, Nixon works to extend the qualitative framework with quantitative concepts like workload statistics [Nix94a], which are used as arguments for selecting among implementation alternatives. This integration is, together with applying the framework to extensive portions of larger systems, an area for further research.

5.6 Chapter Summary

This chapter has presented current methods for performance engineering of information systems. The SPE approach is embedded in the waterfall lifecycle model, and will therefore benefit from the extensions of the waterfall lifecycle like prototyping and iterative development. However, the SPE approach does not elaborate on the integration with conceptual modelling. Therefore, the focus on the developer is not as good as in the PPP/SP method. The work concept which leads to static modelling is unique to SP. More practical experience is needed with PPP/SP. The COMPLEMENT framework is fairly similar to the PPP/SP framework, but the focus on static modelling seems to be missing. A HIT + PPP or a LQM + PPP/SP solution would be interesting to elaborate further on. The integration of quantitative

and qualitative modelling also seems promising.

Chapter 6

Method for Performance Engineering of Information Systems

The general method for performance engineering of information systems is presented at an abstract level in this chapter. The focus of this presentation is on the computerised part of the framework which was introduced in Figure 2.4, i.e. the framework in Figure 4.3. The method is applied to the Blood Bank Case Study in Chapter 7. The organisational parts of the framework are introduced in Chapter 9. The same method is then applied to the organisational and computer parts of the Gas Sales Telex Administration Case Study in Chapter 10 and 11 respectively. This method was derived from all these case studies, but primarily from the Blood Bank Case Study. Even though this *method* is meant to be general, more efforts have been directed to some areas of performance engineering than to others. The boundaries of the *discussion* are:

Design of a complete system or an application Several types of performance engineering problem can be solved with this method: (1) The whole system is to be designed from scratch; (2) some of the modules may be available prior to the design of a system and the system is developed on top of this platform; (3) the system is designed, but should be installed in a new environment. This framework is useful for all of these situations. The case studies which are used to illustrate the method are of type (2), i.e. with focus on design of applications on top of an existing platform. The scope of the discussion could also be extended to cope with the design and operation of several applications on several different platforms. Design of platform components could also be considered more extensively.

First phases in method Phase IIB., III. and IV. in the method in Table 6.1 are not considered to the same extent as the first phases in the method. The focus of this presentation is on the first phases in system development, where

the focus is on static modelling. In later phases, more accurate dynamic performance models are needed, e.g. a resource may be modelled progressively more detailed, e.g. dynamic modelling replacing static modelling, single-class queueing networks replacing bounds analysis or simulation replacing multi-class queueing networks.¹ Dynamic modelling is extensively described in the literature, e.g. [FSZ83, LZGS84, Kan92, Smi90], and is not focused in this thesis.

Limits on validation In the ideal case, case studies which are used to validate this method should influence the resulting performance of the ongoing projects. However, it was not possible to give performance guidance in the two case studies, because:

- Performance engineering depends on formality in information system development. Both in the Blood Bank Case Study and in the Gas Sales Telex Administration Case Study, process models (DFD models) had to be built from scratch before performance engineering could start. If existing process models could have been extended with relevant parameters, understanding of the system would be easier to get, and the cost of performance engineering would decrease.
- It is hard to introduce new methods in an organisation. Skills and competence are missing.
- Need for manpower resources. Since this is a new method, it is hard to get access to sufficient manpower resources.

These difficulties will be quite analogous to the difficulties when introducing CASE-tools in an organisation [Par90]. Performance engineering would have been simpler if performance models of the platform had been available. Performance modelling of the platform was a complex task, which is described in more detail by Vetland [Vet93].

The first part of this chapter is prescriptive and presents an overview of the method in Section 6.1. In this section, all the phases in the method are also described. The second part of this chapter focuses on specific problems and is more descriptive. This second part consists of all the remaining sections in this chapter. Section 6.2 describes the world concept which is an important part of the method. In this section, the hierarchy in SP is also compared with the hierarchy of DFD. Section 6.3 comments on the rationality of the method. Section 6.4 comments on the interaction with the method and object-oriented development methods. Finally, Section 6.5 compares the method with the SPE method. Part of the developed method is presented in [BOVS96]. Inspiration from this joint work with Opdahl and Vetland will be evident in this chapter. Parts of a performance engineering method is also implicitly presented in Opdahl's thesis [Opd92], where method issues were not explicitly considered.

¹The IMSE project investigated how this shift in modelling approach could be done smoothly [Hug89].

<u>Overall method</u>
<p>I. Specify system requirements:</p> <ul style="list-style-type: none">IA. Determine objectives of performance model.IB. Specify system boundaries.IC. Estimate workload.ID. Determine performance requirements.
<p>II. Create performance model:</p> <ul style="list-style-type: none">IIA. Establish static model.IIB. If necessary, establish dynamic model.
<p>III. Guide system development.</p>
<p>IV. Verify and refine overall performance model.</p>

Table 6.1: Overall method.

6.1 Method Overview

To introduce the method, a simple, non-iterative view of the method is shown in Table 6.1. Iterative refinement of performance models is an important part of the method. Sections 6.1.2 through Section 6.1.8 in the chapter is an elaboration of the method in Table 6.1.

The performance model should reflect the performance of the system. The system may of course also be affected by the method. If inadequate performance could be isolated to one part of the system, then the performance in this part should be improved. In cases where potential improvements are infeasible, for example because of excessive costs, performance requirements must be relaxed or the overall system must be changed. This illustrates that performance engineering also involves a tradeoff with factors outside of modelling, e.g. cost, as described in Section 5.3.1. The relation between the method for performance engineering and system development is explained in Section 6.1.1.

This method makes a distinction between modelling of projected applications and existing subsystems. All resources in a system may be divided into existing resources and projected (or new) resources. *Existing* resources can be measured, whereas *projected* applications cannot be measured. A projected resource can only be modelled. The existing resources may serve as a platform for the projected resources.

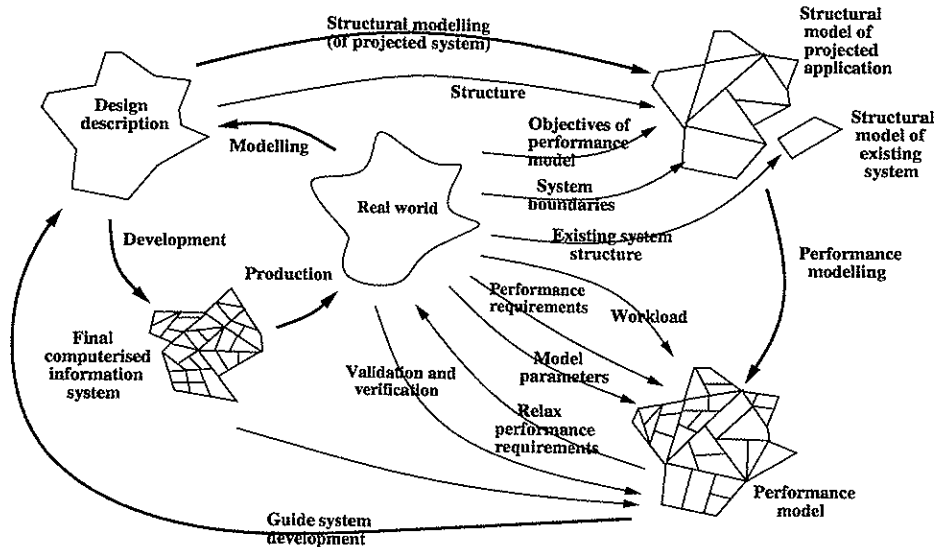


Figure 6.1: Relation between information system development and performance engineering.

6.1.1 Relation to Information System Development

The method for performance engineering of information systems is embedded into a wider method for information system development as shown in Figure 6.1. The bold lines in the figure shows the main flow of control in the figure, while the solid thin lines are mainly used for feedback information flows. The left, inner, bold-lined loop in this figure is for information system development and is a condensed version of the PPP method in Figure 5.4. The outer, bold-lined loop shows the performance modelling, and its interaction with information system development in the inner loop. Each graphical symbol in the figure is explained below:

Real world The relevant part of the system prior to development (c.f. real world system in PPP method in Figure 5.4).

Design description A model of the real world and in particular of the computerised system. Typical models are process models and data models (PrM and PhM in PPP as described in Appendix B).

Final computerised system A working computerised implementation of user requirements (c.f. PPP method in Figure 5.4). This system is put into production in the real world.

Structural model of projected application, an outline of a performance model of the projected application. The performance engineering process starts when performance is identified as a risk factor in information system development

(c.f. the spiral model in Section 5.2.5). The process ends when the system has acceptable performance or when it has been concluded that it is not possible to build the system with acceptable performance. Based on objectives of the performance model (described in Section 6.1.2), performance model boundaries are determined (in Section 6.1.3). The components in the structural model are based on structure information from the design description.

Structural model of existing system, an outline of a performance model of the existing system is made based on the real-world system. The existing system will often form the platform for the projected system.

Overall performance model, a progressively more refined performance model with parameters consisting of both a static (see Chapter 4 and Section 5.4.5.1, and 5.4.5.2) and (if necessary) dynamic models (see Section 3.2). Static model parameters consist of complexity specifications for the components (described in Section 4.2.1.1), compactness specifications (see Section 4.6.1) and resource demands (see Section 4.2.4) for the primary components. If needed, dynamic models may be made for the existing system or for the overall system, consisting of both the existing system and the projected application, to increase the accuracy of the performance model. Relevant parameters for (dynamic) queueing network models are described in Section 3.2.1 and in [LZGS84].

Based on the real world the workload is estimated (in Section 6.1.4) together with performance requirements (in Section 6.1.5). Based on model parameters, the performance model predicts system performance and guides system development. Part of the model may be verified by parts of the final computerised information system during development. After the projected system is put in production, it is possible to verify the projections of the performance model and the workload model.

Since modelling and design of information systems is an iterative process, there will be several iterations in the outer loop in Figure 6.1. When a new information system model is designed or considerably changed, a new turn in the loop is performed if performance is a risk factor in this phase. As soon as parts of the information system are finished, verification of the performance (sub)models is performed, using instrumentation in the finished parts. The performance model in Figure 6.1 guides system development, for example with work budgets, as described in Section 6.1.7. Performance requirements are relaxed if the initial performance requirements are impossible to fulfil. When the information system is finished, it is possible to verify the predictions in the performance model as described in Section 3.3. In addition to the non-functional requirement performance, all the other functional and non-functional requirements should also be met during development. This tradeoff process is described in general in Section 5.1 for non-functional requirements and in [LSS94] for functional requirements. When doing performance engineering, the most important factors in this tradeoff are, however, cost, accuracy and benefits as described in Section 5.3.1. During the implementation phase in the information system lifecycle, hardware components are selected subject to constraints by earlier choices [Fer78].

6.1.2 Determine Objectives of Performance Model (IA.)

By means of the method in Table 6.1, the system S , the workload W and the performance requirements P are described with models. The purpose of a performance model is to facilitate in making a system S which satisfies performance requirements P under a certain workload W . Therefore, the performance model should be geared towards identifying at an early design stage potential difficulties in reaching this objective. Identification of the objectives of the performance model is the focal point of all the later phases in the performance engineering method. Examples of overall objectives of the performance model are:

- Make sure system S meets the performance requirement P with the workload W . This is a standard design problem as formulated in Section 3.1 by the function $P(S, W) \geq p$, where p is the performance requirements.
- Find the bottleneck in system S for a given workload W and performance requirement P .
- Find the cheapest (e.g. cost-optimised) system S for a given workload W and performance requirement P .

The latter objective depends on an extended method where cost also is integrated, and is outside the scope of this thesis.

6.1.3 Determine System Boundaries (IB.)

Before modelling starts, it is important to determine the boundaries of the system and therefore also of the model. As stated in Section 3.1, both the workload and the performance heavily depend on the system boundaries. A system (and a subsystem) has at least one systemic property not possessed by any of its parts [FRI95]. The definition of subsystem in the context of performance in Section 4.3.1 imposes further restrictions on valid system (and subsystem) boundaries. Thus, not all boundaries which are possible correspond to a system boundary. See Section 6.2 for a discussion of the “world” concept which is relevant for system boundaries.

6.1.4 Estimate Workload (IC.)

The boundary between the environment and the system determines the operations which make up the workload. As stated in Section 4.5, these operations are specified for example as:

$O_{Secretary_memory} = [get_internal_address, get_external_address, store_achieve, get_achieve]$

As discussed in Section 4.2, a *workload specification* W will contain a description of both work and load. For an open system of the transaction type with one class, the workload specification is:

$$W = \vec{\lambda}$$

Most organisations keep workload statistics at the organisational level already (this was for example the case in the Blood Bank). Therefore, it is often only a matter of compilation of statistics, i.e. talking to the right people in the organisation. Because of the several local realities (c.f. Section 8.1.2) in an organisation, people do not know about all the material which is available. Therefore, for an organisation to see all numbers compiled in one place is in itself instrumental to give overview which will provide new insights. This is the process of externalisation of local realities into the organisational reality as elaborated in Section 8.1.2.

The workload will often consist of several parts. One part of the workload may be well-defined and more or less periodic while other parts will be inherently ad hoc, i.e. per definition impossible to predict. For periodic workloads, the notion of busy interval is useful for identification of critical intervals [Fer78]. The busy interval is the time interval with the highest workload for the use of a resource r , and is specified as:

$$I_r = (T, t_s)$$

T is the length of the interval, and t_s is the start time. If the start time t_s is not fully specified, as e.g. 1.00 PM, but not which day, it will implicitly refer to a period. In this case, this period is one day, but it could also be a week or even a year.

Often, the interval duration is one hour. As an example, in telecommunication each day has a *busy hour*. The day is divided into 96 quarters and the four succeeding quarters with the highest load make up the busy hour. The *time consistent busy hour* is defined as the four succeeding quarters which have the highest load during a longer period than one day. This period could be one week, one month or one year. The busy hour will often have a higher load than the time consistent busy hour, but *on the average*, the time consistent busy hour constitutes the largest load on the system.

In the general case, each resource and each workload will have individual busy intervals, because bottlenecks depend on a low-level definition of work. The CPU may for example have another busy interval than the disk. For computerised resources, it is known that small variations in processes that run on the system may change the bottleneck. This makes it hard to make general performance models.

Estimation of the busy interval may be problematic (1) when the distribution of workload during the busy interval is uneven, e.g. all the traffic within a fraction of the interval [Buz86]; (2) with infrequent operations which may create a lot of problems when they occur. Even if their average workload is low, the work they devolve on the system may be substantial.

6.1.5 Determine Performance Requirements (ID.)

Performance requirements for operations with high workload have to be characterised more carefully than less critical operations. Performance requirements should be specified in terms of performance metrics, which normally is response time or throughput for the operations in the system (see Section 3.2.1 for other performance metrics). For an open system with one class, performance requirements will be a vector of response time, R , for all operations: ²

$$P = R$$

Often enough information is not available to set the performance goals, as discussed in Section 6.3. In these cases, it is possible to defer setting performance requirements, and (later) ask the customers if the *estimated* response times (based on a performance model) are satisfactory. Lack of precise performance requirements may disturb the notion of a (performance) contract between developers and customers. A weak (performance) contract is normally to the disadvantage of the customer. Developers may excuse inadequate performance by blaming the customer for poor specification of performance requirements or poor workload specification.

Phase IIA: Establish static model

- IIA-1. Specify model components.
- IIA-2. Parameterise static model components of projected application.
- IIA-3. Parameterise and validate static model components of existing system.
- IIA-4. Evaluate and verify static model.

Table 6.2: Static model method.

²See Section 4.3.2 for performance requirements for closed systems.

6.1.6 Create static model (IIA.)

In this phase, the static model is constructed. This phase contains several (sub)phases which is shown in Table 6.2. As described in Section 6.1.1, components form the basis for the static model, as described in phase IIA-1. These model components are derived from the design description. In phase IIA-2, the static model of the projected (new) application is parameterised. These phases are considered more extensively in Section 6.1.6.1. Phase IIA-3. contains the parameterisation of the static model of the existing system, which is considered in Section 6.1.6.2. In phase IIA-4. the complete static model is evaluated and verified. The static model is evaluated by including resource demands as described in Section 4.2.4. With the introduction of resource demands, it is possible to predict response time without contention as illustrated in the Gas Sales Telex Administration Case Study in Section 10.3. Verification is described in Section 6.1.8.

6.1.6.1 Modelling of Projected Application (IIA-2.)

The projected components are modelled based on design descriptions which are annotated with relevant parameters as explained in Section 5.4.5.1. As depicted in Figure 6.2, several roles participate with parameters.

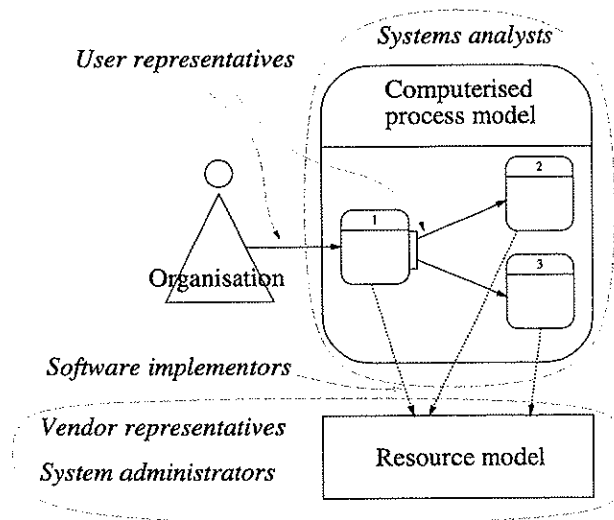


Figure 6.2: Different roles give different types of parameters as illustrated by the broken lines. Roles are *emphasised*.

The parameters from each role are described below:

Systems analysts make design descriptions which reflect the problem area. In Figure 6.2 it is depicted how systems analysts make PrM models which reflect information processes and the flow of information and control between these processes.

Software implementors annotate the PPP models with resource consumption for every process in system-independent form. They will also identify the branchings in the system and correlate them with branchings at the organisational level.

Vendor representatives give basic computer system information and help specify the target platform.

System administrators contribute with an overview of site-specific features of the computer system, e.g. operating system parameters.

User representatives give estimated workload, and the branching probabilities for the branchings which were identified above. "User representatives" do not necessarily mean end users, but may also mean managers with an overview knowledge of the workload on the application. Capture of parameters in the organisation is described in more detail in [ST89]. Forecasting may also be necessary if there is an expected growth in the workload [CMBN82].

Of course, several roles may be solved by the same person. These roles are linked to the world concept in Section 6.2. The systems analyst is for example not in the same world as the software implementor. Vendor representatives and system administrators contribute to the existing performance model which is described in Section 6.1.6.2.

6.1.6.2 Modelling of Existing System (IIA-3.)

When the design description starts to be a basis for implementation, a software/hardware platform needs to be selected. Platform components will often *exist* before the development project starts. In general, the problem of selecting a software/hardware platform is hard, because a large range of alternatives have to be analysed and because vendor data often are valid only under narrow conditions [BF87b]. Failure to recognise model limitations is one of the biggest shortcomings of performance engineering in the development lifecycle [Fox89]. Inaccuracy in the resource consumption estimates together with inaccuracies in the existing system performance model are the two main sources of error in the modelling process [Fox89]. Making a model of an existing system is a highly iterative process, where model formulation, measurements, documentation reading and communication with other specialists are highly intertwined. Because of its complexity, such modelling will always have a creative element. However, if methods like SP become commercially available they will give

good assistance. Modelling of existing systems with SP is described in Vetland's thesis [Vet93]. For availability of complexity specifications in SP it is possible to distinguish between the following situations:

Used in the organisation Since the software is in production use in the organisation, it is easy to measure and therefore to get complexity specifications.

In the market but not used in the organisation. This means that complexity specifications are available, but either some other organisation's software must be measured, or complexity specifications must be purchased.

Under development in the organisation, which means that it is possible to find approximate complexity specifications like in the Blood Bank Case Study.

Under development in market In this situation, it is often hard to get access to enough information. Approximate complexity specifications may be available from the vendor.

Needed but no product yet Almost impossible to get complexity specifications, since only requirements are available.

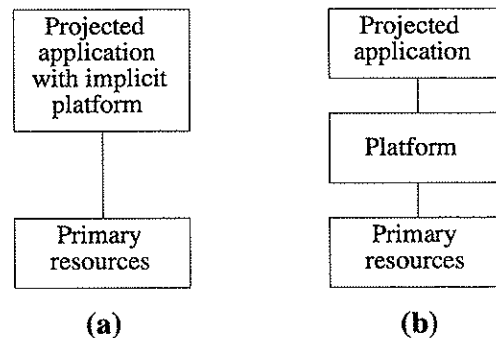


Figure 6.3: In (a) the platform model is inside of the application model, whereas in (b) the platform model is outside of the application model.

With an explicit platform model of the existing platform as shown in Figure 6.3 (b), processes are annotated in terms of the work on the platform model, and the performance information in the platform world is hidden. Thus, performance engineering is easier in Figure 6.3 (b). This is in contrast to the situation in Figure 6.3 (a) where there is no platform model. In (a), the developer has to know all the details inside of the platform world in order to annotate his application processes in terms of the resource demands for the primary resources.

The target platform model is an ADT (abstract data type) as described in Section 4.5, and offers a set of operations. The specification of a platform ADT is a matter of workload characterisation, where the basic question is on selecting representative operations for the platform system. As often during modelling, this is

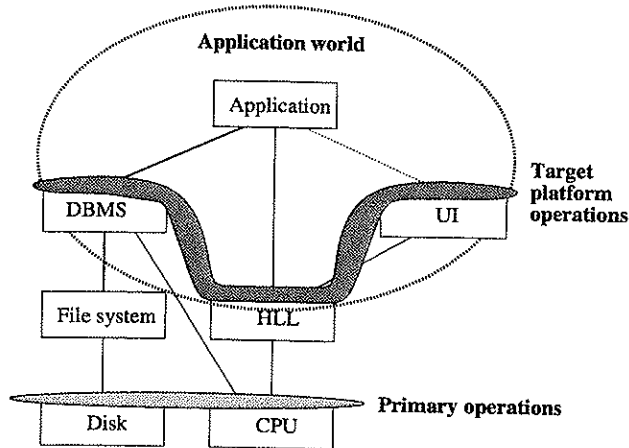


Figure 6.4: An example of a platform model in the Blood Bank Case Study.

some kind of a chicken-and-egg problem, which can only be solved with the feedback provided by iteration and verification. If experience from similar projects exists, previous iterations are available. If no experience is available, iterations must be performed within the project itself. The high-level platform operations should be close to the operations perceived by the projected system developers. To further ease the effort required in the annotation process, few parameters should be asked for, i.e. a very representative platform characterisation may be too complicated to be used by the developers.

In Figure 6.4, an example of a target platform for the Blood Bank Case Study is shown. As indicated in this figure, the application and the ADT (abstract data type) of the DBMS, HLL (high-level language) and UI (user interface) is part of the application world. The primary operations are also shown in the figure. In this case, the primary resources are the disk and the CPU (this figure neglects the network which was also a part of the Blood Bank Case Study).

6.1.7 Guide System Development (III.)

The core of the interaction between the performance model and the design description is exchange of work estimates/measurements and work budgets as illustrated in Figure 6.5. Development of information systems with performance engineering is like ordinary budgeting/accounting applied to work and structure during development of an information system. The non-linear response time curve in Figure 3.6 illustrates why it may give inaccurate budgets by just adding response times for the same resource. Response time is a performance measure. Since performance measures in contrast to work measures are not additive, budgets in terms of performance measures are harder to handle than work budgets.

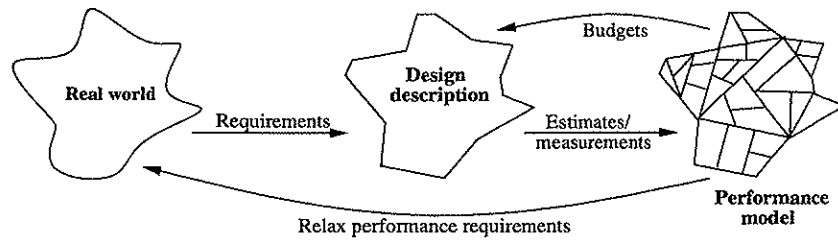


Figure 6.5: Budgets guide the design process. If the budgets predict inadequate performance, this may be cured by refinements in the design description, or relaxed (performance) requirements. The names on the links in this figure are more detailed than to the link names in Figure 6.1, e.g. the link “Guide system development” in Figure 6.1 will at least contain the link “Budgets” in this figure.

The best estimates from the developers are used when the performance model is made. Based on the validated overall performance model, it is possible to derive work budgets for each module of the system. When designing a module, the complexity specification for this module will serve as a budget for each operation in the module. Similarly, space budgets are expressed in terms of extent and compactness specifications.

During development of each operation, work budgets for operations may overrun. This overrun may be determined with a new iteration of the performance model. Updated parameters from the modified design description give a modified performance model which gives new performance predictions. If inadequate performance will be the result of this budget overrun, the design description must be modified, or its implementation must be improved. If it is impossible to sufficiently improve the projected performance by changes in its design or its implementation, the performance requirements (or some other requirements) must be relaxed.

Estimates are replaced by measurements as input to the performance model as soon as possible. Instrumentation is needed to verify the performance model, and should be planned during design when it is possible to establish a direct correspondence between user functions and measurement events. It is more easy to maintain this relationship than to reconstruct it later[Smi90].

6.1.8 Verify and Refine Overall Performance Model (IV.)

The overall performance *model* can only be compared with the implemented *system* when the final computerised information system runs in production. During production, the system is running under real workload. It is then possible to see if the estimated organisational workload corresponds to the real organisational workload. This comparison is termed verification as described in Section 3.3.

As *components* of the projected system are implemented, they can be measured and verified. Measurement data are also useful for testing, giving an early validation of estimates. Measurements should replace estimates as early as possible. Measurements will often depend on instrumentation [Fer78]. Validation and verification will be done on several levels. During parameter capture, it may be possible to compare information from several sources as in the Blood Bank Case Study in Section 7.1.4.1. As long as the model is small, it is easier to spot the parameters which must be corrected, or to introduce more sophisticated modelling techniques (contention modelling, simulation instead of queueing networks) if accuracy is not good enough and if cost is not exceeded. Sensitivity techniques make it possible to focus on the important parameters [Opd92]. If the whole system is not finished, an artificial environment must be used for the unfinished parts of the system. This artificial environment must simulate a production environment. With this verification, feedback on resource budgets could be given. Validation and verification is when modelling meets practice, and is an important learning process for the people involved.

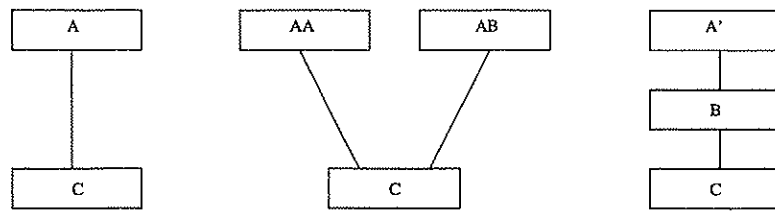
6.2 Worlds

The concept of a “world” in an information system refers to an information layer in an information system. This information layer may be hardware resources, software resources or organisational resources as briefly introduced in Section 2.2 and elaborated in Chapter 9, e.g. Figure 6.4 shows the world of the blood bank application developers. The world concept was an important part in our learning process. With this concept, suddenly the coin dropped in the box of understanding.

Worlds in information systems are connected to the concept of hierarchical decomposition and resource platforms in SP, which is described in Section 6.2.1. The concept of worlds is wider than the concept of *views* in SQL databases [Dat86] and in PPP (c.f. Appendix B). For DBMSs, the world concept is illustrated with this quote [Dat86, p. 48]:

(T)he DBMS has a view of the database as a collection of stored records, and that view is supported by the file manager; the file manager, in turn, has a view of the database as a collection of pages, and that view is supported by the disk manager; and the disk manager has a view of the disk “as it really is”.

Section 6.2.1 compares hierarchical decomposition in SP with the hierarchical decomposition in DFDs (data flow diagrams), and is important for an understanding of the world concept. Section 6.2.2 explains how worlds are used during development of information systems and also describes the connections between worlds and performance engineering.



a) The module before decomposition. b) The module after DFD decomposition. c) The module after SP decomposition.

Figure 6.6: Decomposition of a module in the SP language and the DFD language.

6.2.1 Hierarchical Decomposition

The level of abstraction in SP is not the same as the level of abstraction in a data flow diagram (DFD). This is part of the reason why SP is hard to use for newcomers. SP models software and hardware resources at several levels of abstraction. The focus is not so much on modelling of processes. While DFD decomposes the processes as shown in Figure 6.6 (a), SP decomposes the resource platform as depicted in Figure 6.6 (b). In DFD, it is also possible to work on several resource platform levels, but this is not explicit, as in SP. Thus, the decomposition concept in SP may be viewed as orthogonal to the decomposition concept in DFD. SP takes care of resource levels whereas DFD models processes in detail. Within an SP module, there may be several DFD processes. SP may be viewed as a “macro” modelling language which shows the overall structure, while DFD is a “micro” modelling language where details are illustrated. As an example of SP decomposition, the SP module `Telex_secretary` in Figure 10.4 in Section 10.1.3.2 is decomposed in Figure 10.7 in Section 10.2. Accordingly, Figure 10.7 may be aggregated to form the `Telex_secretary` in Figure 10.4. In DFD it is also possible to make process hierarchies,³ but since the level of detail in DFD is larger than in SP, these process hierarchies will in practice become very large. Overview is therefore easily lost.

PrM is an extended DFD language as explained in Section B.1.2. When translating PrM diagrams to SP diagrams, triggering flows in PrM will often correspond to operations in SP [Opd92]. The typing of operation in SP (processing, communication and memory) has similarities with the basic concepts process, link and communication in DFD. Information flows and sequence of operations are lost in SP. This is illustrated in Figure 6.7 and is the price we pay for a more compact representation. Since operations in SP are more compact than processes in PrM, SP diagrams can contain more information in the same diagram. The model for the secretary in Figure 4.2 and the workflow systems in Figure 9.5 are concrete examples. The dotted lines in Figure 6.7 show how each PrM process devolves work in terms of suboperations.

³See Section B.1.2 for a description of process hierarchies in the DFD language PrM.

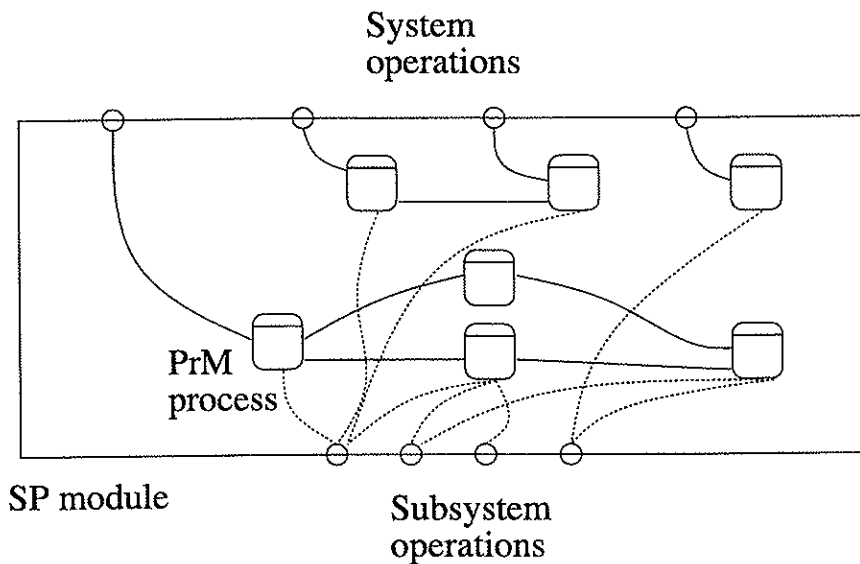


Figure 6.7: There will be several DFD processes within an SP module.

6.2.2 Worlds in Information System Development

The quote in the beginning of Section 6.2 shows that even if developers with different roles work with the same subsystems, they will work in different *worlds*. As another example, the information systems developer is not in the same world as the hardware developer even if both work with hardware systems. Their tasks are different. Whereas the information systems developer *configures* hardware systems, the hardware developer *constructs* hardware systems. Modern systems are complex and offer several operations with the same functionality [Fox89]. How these operations are used by the developers depend on their knowledge, or the world they work in.

A designer should know more than others about the resources in his world. The concept of world is related to the concept of internal realities of organisations [Kro95]. The things you cannot understand is not part of your world. Hence, worlds have to do with information hiding, and is problematic in performance engineering which is holistic: no part of the system can be neglected (c.f. Section 3.1.3).

The *world* concept for information systems and organisations serves to explain why performance engineering is hard to grasp. Every developer of an information processing systems knows about the resources which are inside of his world. These resources use other resources which one may not know about, i.e. they are outside of this world. However, when a performance model is developed, knowledge about all the worlds in the system must be available.

A major problem in performance engineering is therefore a *communication problem* between the developers in each world and the performance engineer who is responsible for the overall performance model. This problem is especially tricky when many design worlds exist, usually during the design phase. Performance requirements are stated by users in cooperation with developers for the application world, i.e. the world which the application developer works in.

Some of the non-functional requirements are hidden from the developers in the application world, but they are relevant for developers in other, lower worlds. Because of complexity, some of these non-functional requirements are temporarily ignored during development. In the end, however, all of them have to be considered. The designer or user often finds it hard to grasp what the performance engineer is aiming at. Often, the performance engineer is satisfied with a much lower level of accuracy than the designer expects. Therefore, the performance engineer must supply designers and users with estimates of the cost of typical functions they should use [Bel87].

Recursive applications of the abstract virtual machine in SP takes you into new resource worlds. Processes in a virtual machine or a world can only use resources which are available in this world. Data structures are constructed in the world where the design takes place (usually only the top level). At every other level there is also some degree of freedom. For example in a database, it is possible to change the concrete data structures.

6.3 Rationality of Design Process

The method is presented as a series of iterated phases. Even when this method is formulated as a series of phases, an *overview* is needed before this method can be applied with success. The more overview a performance engineer or an system developer has, the better. With overview, it is possible to focus on the important tasks at each stage of performance engineering. Hence, the risk identification in the spiral model in Section 5.2.5 is an example of overview, now covering the total system development and not only performance engineering.

Often we have the situation in performance engineering where the information is not available. For example, the busy hour will depend on the workload at the primary resource level which is only available in phase IIA. or even IIB. These chicken-and-egg problems may be solved by iteration, i.e. by improving the estimation of the busy hour in the next iteration of the method. Let us call this brute force application of the method for a "rigid" method. A rigid method is easy in theory, but in practice it will often be too costly.

In contrast to the rigid method, we may use tacit knowledge about performance and performance problems. Tacit knowledge cannot be directly represented, but shows up in actions of the person having the knowledge [SK93]. Some of the tacit knowledge will be specific system knowledge. Specific system knowledge will always

beat general system knowledge. A person with specific system knowledge will know the important aspects of design. Examples of specific system knowledge is familiarity with the operations typically used on the platform, and typical platform level resource demands for application level operations. The tacit knowledge or intuition from the domain expert should be supported with accurate tools and methods. Success without specific knowledge can be achieved through the rigid method. The Blood Bank Case Study is partly an example of this. Hence, domain expertise is to some extent replaceable by raw manpower and method iterations. For example, if a designer knows from experience that the server used to be the bottleneck in the type of applications you make, there is no need to investigate consumption of client resources, i.e. the parameter estimation effort can be reduced. There are special problems on projects with no previous experience where domain experts are badly needed but often not available.

In practice, a synthesis of the intuition-based ad-hoc method and the rigid method will be used in an interactive process. The engineering part is to keep uncertainty under control. This is in line with common observations: a rational design process is not (completely) followed in practice, but it is important to *try* to follow it [PC86].

6.4 Object-oriented Performance Engineering?

This method is formulated in the context of traditional development methods. Recently, object-oriented (OO) methods have become more widespread. We now consider how our approach could be applied in an OO setting. The phases in the method are in principle independent of development methods. Using an OO development method will only change the interface between the design description and the performance model. It is therefore sufficient to consider the relation between OO models and the basic framework of this thesis. Section 6.4.1 describes the basic concepts of OO in a fairly standard fashion. In Section 6.4.2 the common OO method OMT is used as an example.

The origins of object-orientation can be traced back to the Norwegian simulation language SIMULA (c.f. Section 3.2) in the 1960s. Even today, OO is most widespread for programming (OOP, Object-Orientation Programming). OO Design (OOD) and OO Analysis (OOA) are also used commercially, but to a less extent. There is no clear separation between OOD and OOA, which has been referred to as the seamless transition from OOA to OOD. The object-orientation paradigm is basically a bottom-up approach to software development, since the OOA focuses on the application domain objects [SK93].

6.4.1 Basic OO concepts

The basic concepts of object-orientation are objects, classes, encapsulation, inheritance, polymorphism, and dynamic binding [SK93]. An *object* is an *encapsulation* of a set of operations (methods) which can be invoked externally and of a local state which remember the effects of the operations [BGHS91]. The value of the local state can only be accessed by sending a message to the object which calls one of its operations. An object has a unique and unchangeable identifier.

Objects are organised in *classes*, which provide the implementation of operations. Classes are again organised in generalisation hierarchies, where it is possible to *inherit* operations. Objects and classes are also organised in aggregation hierarchies so that composite objects or classes can be made based on more primitive objects and classes. An abstract interface which shows only available operations and no implementation may be called a *type*. It is possible to construct type hierarchies. Since several classes may implement the same type and one class may be implement several types, there is not necessarily a structural relationship between class hierarchies and type hierarchies [OSV93].

A *polymorphic* operation can be applied uniformly to a variety of objects, i.e. the same *draw* operation may draw different graphical objects, such as arcs, rectangles, circles etc. The algorithm to be used for drawing may be selected at run time, which is termed *dynamic binding*. Dynamic binding is an effective mechanism to implement polymorphism.

6.4.2 OMT

To be more specific, the Object Modelling Technique (OMT) will be used as an example. OMT is based on three major models [SK93]:

Object model describing the static structure of the objects and their relationships. The object model and may be seen as an ER diagram extended with for example generalisation hierarchies from semantic networks.

Dynamic model representing the state transitions of the system. The dynamic models are based on concurrent state-transition diagrams extended with constructs similar to those found in Statecharts [Har88].

Functional model describing the transformation of data within a system, using DFDs (Data Flow Diagrams).

OMT distinguishes between eight phases of system development [SK93]. The first five phases focus on the object model and therefore on the static aspects of development. An important issue here is to identify the objects. Objects or combinations

of objects could be characterised for performance engineering purposes by SP components.

Phase six in the OMT system development, **Verifying that access paths exist for likely queries** focuses on dynamics. This phase will be assisted by a combination of dynamic and functional models. PrM, which is used in this thesis, is a superset of DFD. Since OMT uses DFD, the framework in this thesis could be applied. Based on a functional description of the interaction between objects, complexity specifications may be derived. Some of the additional constructs in PrM are necessary if a performance model should be made during design, e.g. branching probabilities. As described in Section 5.4.5.1 workload intensities and resource demands are also needed.

The two last phases in OMT are **Iterating and refining the model** and **Grouping classes into modules**. While **Iterating and refining the model** will literally refine the models, the phase **Grouping classes into modules** will determine the final SP structure.

6.5 Comparison with Current Best Practice

The SPE method which is described in Section 5.4.1 is one of the most well-established methods for performance engineering of information systems, and also represents current best practice. If we compare the SPE method with the PPP/SP method in this thesis, each activity in the SPE method corresponds to an activity in the PPP/SP method:

Define SPE assessments for lifecycle phase Determine objectives of performance model. Specify System Boundaries.

Create concept for lifecycle product Develop design description.

Gather data Estimate workload. Determine performance requirements.

Construct & evaluate appropriate model Establish projected and existing components of the performance model. Parameterise these submodels.

Report results Guide system development.

Alternatives preferable Changes in the design description may lead to a new turn in the outer loop in Figure 6.1 and changes in the performance model, or performance requirements may be relaxed.

Complete lifecycle product Develop design description based on feedback from the performance engineering process.

Verification & Validation Validate submodels and verify projections from the overall performance model. ⁴

Enter next phase Refine the design description or implement part of the final computerised information system.

Our method extends the SPE method in certain respects:

Design description Easier interaction between the design description and the performance model. This is demonstrated by annotating PrM models with performance parameters, for example in the Blood Bank Case Study. The resource concept in SP complements the process models and data models of common CASE-tools, in this case the experimental CASE-tool PPP [GLW91]. Work budgets are also more convenient during development than budgets in terms of performance (c.f. Section 5.4.6).

World concept In SP the world concept is made explicit, deepening the understanding of performance for large systems. The method is based on this “world” concept. Two worlds of competence are needed in performance engineering. Information system developers know details about the developed application (but have often only superficial overview of the hardware and software platform), and system architects have an overview of the total system (including hardware and software platform, but often only superficial overview of the application). Information system developers, with low performance engineering competence, but with high information system competence, could assist in doing performance engineering in the early phases of systems development. In particular, such personnel could be instrumental in resource annotation before professional performance engineers (e.g. systems architects) eventually take over and detail the performance models based on the most important parameters.

Hierarchies In SP it is easier to build hierarchies at a meaningful level of abstraction (c.f. Section 6.2.1). Explicit platforms in SP is an example of a very useful hierarchy level. Together with the hierarchies, the typing in SP (c.f. Section 5.4.6) makes reuse more likely [Vet93].

Organisations It is quite easy to extend the PPP/SP method to organisations as described in this thesis. For an information system, the focus on the organisation is more important than for a software system. Therefore, this is an information system performance engineering method and not only a software performance engineering method.

⁴Smith uses the term verification and validation differently from this thesis. In her view, checking input parameters is verification and checking model projections is validation, no matter if the model is existing or projected. Smith’s view corresponds to Boehm’s view of verification as answering the question “Are we building the product right?”, and validation as answering the question “Are we building the right product” [Boe81]. The meaning of verification and validation in this thesis is described in Section 3.3 and builds on [LZGS84].

6.6 Chapter Summary

This chapter has presented a method for performance engineering of information systems based on the SP language which was presented in Chapter 4. An overview of the method was shown in Table 6.1 and in Figure 6.1. The world concept, which forms a basic part of this method, was also presented. The hierarchy mechanism in SP differs from the hierarchy mechanism of PrM. While a decomposed PrM process still uses the same resources, the resource platform itself changes during SP decomposition. Similarly, the platform concept in SP was described. Compared with SPE, this method has an improved link to the design description through the structural model.

The method is applied to the Blood Bank Case Study in Chapter 7 and extended to workflow systems in Chapter 9. Then, the method is applied to the organisational and computer parts of the Gas Sales Telex Administration Case study in Chapter 10 and 11 respectively.

Chapter 7

Application to a Transaction-oriented System

The blood bank application was a typical medium-size transaction-oriented information system with database and user interface as key platform components. Early in 1991, the Region Hospital in Trondheim was selected as the site for the case study of projected applications, because (1) good working relations had been established between our group and both the Region Hospital and the computer vendor Tandem; (2) the blood bank was the only application under development at the Region Hospital at that time; (3) the blood bank application was sufficiently large so that it was realistic, and equally well reasonably small so that it was feasible to study within the time constraints of a PhD thesis. The method in Chapter 6 was developed as a result of the Blood Bank Case Study in this chapter (Chapter 7). The description in this chapter, which is a condensed and revised version of the relevant parts of [BO91, BOVS92], is therefore also a practical application of the method in Chapter 6. Three design documents which were made during construction of the blood bank application, together with interviews formed the basis for this case study:

Requirement specification [Twi90b] A textual description of the system with main screen images described as figures. Subordinate screen images were described using natural language.

Functional systems design [Twi90a] This document was platform independent and contained a textual description of processes down to three levels of decomposition. The logical database design in the form of ER models and tentative database schemas was also described.

Computer system design [Twi91] A platform dependent document which contained a physical database description with a complete database schema. All screen images are shown as figures. A test plan for the application was outlined. In this document, performance testing was moved to the programming phase, in line with the standard fix-it-later approach described in Section 5.3.

Note that while conceptual structural modelling was done (with an ER model), conceptual functional modelling (e.g. DFD modelling) was not performed.

An overview of the blood bank organisation and application was given in Section 2.1, describing the necessary background for this case study. This chapter is organised as follows: The context of the case study is described in Section 7.1 where also the original scientific objectives of the case study are listed. Section 7.2 illustrates the modelling of the projected blood bank application, while the (existing) platform is described in Section 7.3. The static performance model of the blood bank application is evaluated in Section 7.4.

7.1 Specify System Requirements

The presentation of this case study is complicated because of the three levels involved: (1) The original case study; (2) practical (historical) validation of the performance of the blood bank application; (3) illustration of the method of this thesis. While Section 7.1.1 focuses on the first activity, the other subsections in this section describe the last two activities.

7.1.1 Scientific Objectives

The initial *scientific objectives* of the original case study in the context of the IMSE ESPRIT II project ¹ was formulated as [BOVS92]: ²

F_1^B : Demonstrate how information system performance engineering and information systems development could be undertaken *using the same modelling tools*.

F_2^B : To investigate the feasibility of specifying average resource demands in terms of higher-level operations.

F_3^B : To investigate to which extent this enables developers without much skill in performance engineering to assist in performing engineering.

When these original scientific objectives are extrapolated, the overall scientific objective becomes: "Find a method for performance engineering of information systems." which is the theme of this whole thesis.

¹The IMSE project was described in Chapter 1.

²Superscript *B* in F_1^B , F_2^B and F_3^B refers to the Blood Bank Case Study to avoid confusion with the Gas Sales Telex Administration Case Study.

7.1.2 Objectives of Performance Model

Two objectives were identified for the blood bank application before the case study started.³ The objectives were to investigate if:

O_1^B : The blood bank application has acceptable performance.

O_2^B : The blood bank application will not give a heavy workload on the rest of the hospital computer system.

These “practical” objectives fit well within the “scientific” objectives in Section 7.1.1.

7.1.3 Determine System Boundaries

The blood bank application interacted with other applications, namely the clinical laboratory application and the microbiology application for requesting blood products and performing tests on blood. In addition, an existing security application in the patient administration application was also used by the blood bank as part of the application platform. All these applications are shown in Figure 7.1.

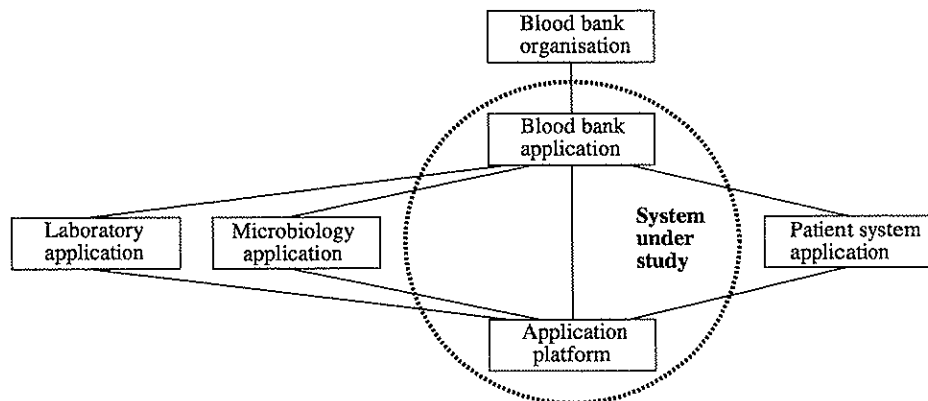


Figure 7.1: An overview of the workload and resources used by the blood bank application.

³No detailed performance evaluation was performed in the original documentation of the blood bank application. In the original documentation [Twi90a], the workload of the blood bank application is superficially compared to other applications, and it is concluded that the workload on the Tandem computer will be smaller than on other well-known architectures, because more screen handling is taken care of by client PCs. For the workload on the client PCs, experience from other users indicates that this will not be a problem. This documentation was only available to us, *after* the blood bank was selected as a case study.

In this case study, the blood bank application and the application platform for this application was the focus of the performance engineering and was therefore selected as the *system under study* in Figure 7.1. The blood bank organisation gives the *workload* on this blood bank application.

The blood bank application would increase the workload on the other applications. In order to assess this increase, performance models for the other applications should also be available. However, the workload from the blood bank application to these other applications was considered to be small. On this basis, the simplifications derived from definitions of system boundaries were justified.

Some extra complications are not shown in Figure 7.1. (1) Some of the other applications may use services from the blood bank. (2) An extra part of the blood bank application for handling financial aspects was added late in the design phase, and is not included in the figure. Financial data is collected all the time, and computed monthly. (3) Finally, the whole blood bank organisation is changed approximately once a year as new products arrive which make the blood products more durable. Therefore, the information system will also be modified from time to time. These three complications are not included in the case study, but are commented in the conclusion.

7.1.4 Estimate Workload

In the organisation, 22 *organisation operations* which gave workload contributions on 13 *application operations* [Flø91] were identified. Estimates for the organisational workload were collected from:

Measurements which showed the flow of information and material in the organisation. Such statistics are often available in large organisations, e.g. for the blood bank, the number of donations per year for each class of blood was collected.

Interviews where roles and operations within the organisation were captured (“Who does what?”). The connections between operations were also valuable. In addition, the flow of information and material to and from the external agents were identified. External agents are persons or organisations outside of the blood bank organisation who interact with the blood bank. Flow of information and material within the blood bank were also identified. The information handling related to the flow of material was also investigated.

Unstructured interviews improved our understanding of the operations carried out in the blood bank. The question “What triggers this operation?” led to interesting and constructive answers about the organisation. For detailed and specific questions related to fluctuations of the workload, structured forms were used. Once the questions were clear, it was easy to ask people to estimate work on the system.

In the Blood Bank Case Study, the busy interval for the existing application was used when analysing the projected application. This was in line with the performance objective O_2^B in Section 7.1.2, which stated that the impact on the existing system should be studied. One busy interval was defined: $I_{Normal} = (1 \text{ hour}, 08.00)$ was the highest workload during one hour of normal operation, i.e. 1 hour from 8.00 AM until 9.00 AM. This was identified to be Monday, Tuesday, Wednesday and Thursday in the weeks before Christmas and Easter (when the blood bank had to build up the blood stock before a longer vacation where donors were hard to get) [Flø91].

In addition, a workload scenario was defined for the highest possible workload on the application $W_{High}^{blood.bank}$, i.e. the workload when a major accident created a large demand for blood. This could happen at any time during the day or night [Flø91], and was a typical ad-hoc workload.

The organisational workload for the 13 application operations in the blood bank during the busy interval I_{Normal} is specified as an open system as follows: ⁴

$$W_{Normal}^{T,Blood.bank} = \begin{array}{l} \textit{search_for_donor} \\ \textit{find_donor} \\ \textit{specially_treated} \\ \textit{normal_donation} \\ \textit{change_prognosis} \\ \textit{pack_plasma} \\ \textit{send_plasma} \\ \textit{albumin_deliverance} \\ \textit{product_delivery} \\ \textit{search_for_product} \\ \textit{fetch_patient_for_trans} \\ \textit{get_overview} \\ \textit{fetch_patient_for_lab} \end{array} \begin{array}{l} \left[\begin{array}{l} 8 \\ 59 \\ 1 \\ 2 \\ 1 \\ 2 \\ 0 \\ 0 \\ 0 \\ 4 \\ 2 \\ 1 \\ 0 \end{array} \right] \end{array}$$

The detailed calculation behind these operation arrival rates are shown in [Flø91]. As an example of the calculation behind this figure the organisational workload on the *application* operation *find_donor* ⁵ is explained below. In order to increase the stock of blood before a long Christmas and Easter vacation, 110 as opposed to normally 80 donor appointments were made every day. Assuming that 10 % of the donors did not turn up for their appointments, the number of donations became 100 every day. Donors arrived for 4.5 hours, so 22 donors arrived per hour. This rate of 22 donors per hour corresponds to one operation at the organisational level, namely

⁴Note that this vector is transposed as indicated by the superscript T . Arrival rates were specified in terms of transactions because the blood bank serves a large number of patients and donors. Each donor and each patient use the blood bank infrequently. See 4.2.2 for more information about the different types of operations.

⁵For clarity, SP components and operations are written with the teletype font.

the `perform_donation` *organisational* operation. Apart from the `perform_donation` *organisational* operation, the `make_appointment_for_donation` *organisational* operation also contributed to the `find_donor` application operation with an arrival rate of 37 per hour. Hence, the `find_donor` operation is triggered $22 + 37 = 59$ times during the busy hour.

`search_for_donor` and `find_donor` represent two different types of application operations. As described above, the `find_donor` has a clear relationship with work carried out by the organisation. When a donor turns up for donation or when the laboratory engineers want to make a new appointment with a donor, the operation `find_donor` is invoked. The operation `search_for_donor` also has direct connection to the work carried out by the organisation. This function is for example used when blood for patients with special needs is requested. Also, this function provides an *overview* of the donors in the blood bank. This function is not connected to flow of material in the organisation, but has to do with coordinating future activities. In this way this distinction conforms well with the difference between production workflows and ad-hoc workflows in Section 9.3. The workload for coordination work is harder to predict than the workload for production work. The workload of coordination work depends to a larger extent on the response time of the information system, the learning effort and how easy it is to do without the services of the information system, e.g. instead of looking up in the blood bank database, you can always look in the physical blood bank.

7.1.4.1 Parameter Validating

The organisational workload parameters described in Section 7.1.4 were partly validated by comparing information from different sources. When the same parameters are collected from both statistics and interviews, a validation of the interviews can be made [Flø91]. The number of blood donations can serve as an example. In 1990, 17990 packets of blood were donated to the blood bank. Divided by the 251 working days in 1990, this gives 71.7 packets of blood on an average day. The blood bank personnel estimated that 80 donation appointments were planned during a normal weekday from Monday to Thursday inclusive. Normally, 8 donors did not turn up for donation. Thus, 72 donors visited the blood bank on Monday, Tuesday, Wednesday and Thursday. Every Friday, 60 appointments were made. Assuming that the same fraction of donors did not turn up for donation, 54 donors visited the blood bank on Fridays. No donations were made during weekends and holidays. Thus, the average number of donations made during a typical day was: $\frac{4 \cdot 72 + 54}{5} = 68.4$. Compared with 71.7 as estimated from statistics for an average day, this gives a discrepancy of 5 % only.

7.1.5 Determine Performance Requirements

In line with performance objective O_1^B in Section 7.1.2, performance requirements were developed. For the original blood bank information system development project, the performance requirement was specified implicitly⁶ as a throughput rate of 100 donors during the normal workload $W_{Normal}^{Blood.bank}$ (c.f. Section 7.1.4):

$$P_1^B: X_{Blood.bank} = 100$$

During the high workload $W_{High}^{Blood.bank}$, the number of transfusions will increase, especially transfusions which need specially treated blood [Flø91]. However, the number of donors will not increase, and the throughput rate is therefore the same during the high workload. Simply put, only the work and not the load increases during the high workload. The throughput rate is the same during the high workload. Performance requirements based on performance objective O_2 were not formulated.

A more detailed performance requirement corresponding to performance objective O_1^B could have been specified in terms of response time for each of the 13 application operations, e.g. less than 3 seconds response time for the `find.donor` operation on the average.

7.2 Establish Components: Projected Application

The blood bank was not initially developed using CASE-tools, which made it necessary to make process models before conducting the performance engineering. Only four data flow diagram models at high abstraction level existed when we created our process models. Fløstrand refined our initial process models [Flø91]. We aimed at creating process models which would have been created if application development had been undertaken using a process modelling tool. As a suitable tool for drawing processes annotated with resources demands, we used a prototype version of the PrM (Process Modelling) tool, designed for making performance engineering during design feasible. This tool was developed by Andreas Lothe Opdahl as described in Section 5.4.5.1.

In this section, we will focus on the `fetch info` process which is one of the 34 primitive processes in the blood bank. This `fetch info` process is a decomposition of the `find donor` process which was used as an example in Section 7.1.4.

⁶Talks with a major Norwegian consulting firm revealed that this is common when performance modelling is performed as a part of information system development: no explicit performance requirements are identified. Based on intuitive judgement, the resulting performance from the performance models, is decided as acceptable or not acceptable. See also Section 6.1.5 for a discussion.

7.2.1 Annotating Process Model

This section will describe the annotation of the blood bank model, first with branching probabilities in Section 7.2.1.1 and afterwards with resource demands in Section 7.2.1.3. Before resource demands can be specified, a resource platform must be defined. The resource platform is described in Section 7.2.1.2. Annotation of a process model with branching probabilities and resource demands was explained conceptually in Section 5.4.5.1.

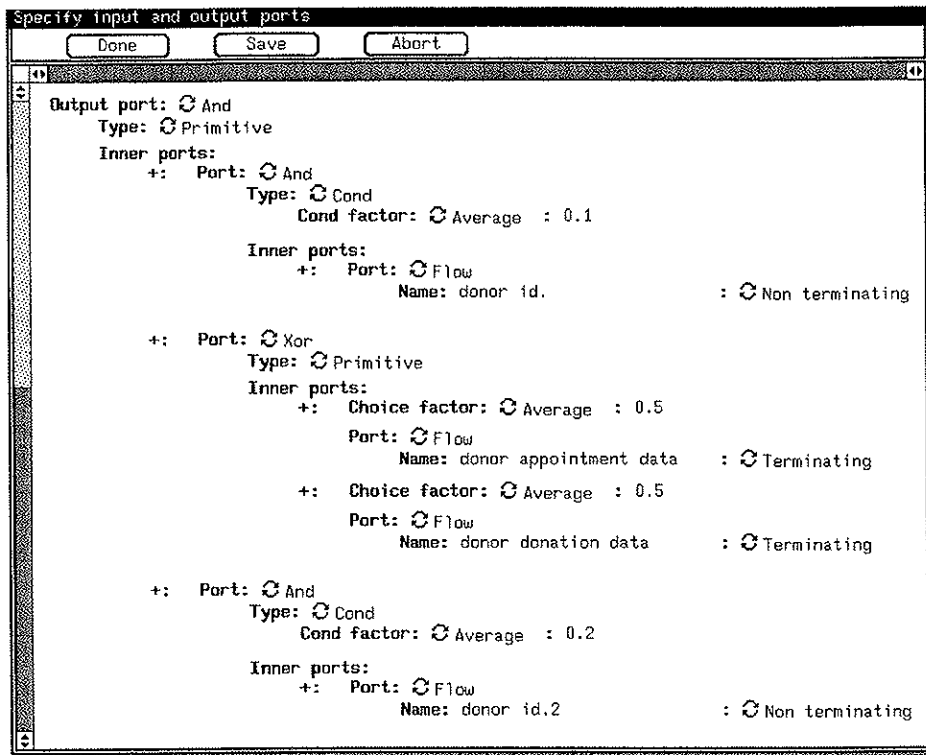


Figure 7.2: PrM output port connectives for the `fetch info` process

7.2.1.1 Branching Probabilities

Figure 7.2 shows how branching probabilities are specified for the PrM process `fetch info`. These branching probabilities came from the users during interviews. The output port⁷ for process `fetch info` has a composite port, consisting of three ports. The first part of this composite port is a conditional port and has the probability of 0.1 for being used. The second part is also a composite port, with two ports, where

⁷For an explanation of “port”, see Section B.1.2 where the PrM process modelling language in PPP is explained.

only one of them will have output with probability 0.5, i.e. one of these processes will have output for each invocation of the process. These two ports are *terminating*, i.e. when data is entered on one of these ports, the process is finished. The last and third part of the composite port is a conditional port, which has the probability of 0.2 of being used. This process is *non-terminating*, which means that the `fetch info` process can continue after output of data from this port.

7.2.1.2 Resource Platform

Database operations were a dominant part of the projected application. Three SQL database operations were considered as important:

select refers to a singleton select where one relation is referenced, and only a small volume of data returned.

select_list is used when a joined, sorted list of tuples is the result of a select. Hence, this operation induces substantially more load than the select operation.

update consists of the three SQL operations update, insert and delete.

These SQL operations use COBOL statements on the TANDEM computer. Therefore, COBOL statements were included as the fourth type of operations on the `server` platform in Figure 7.4.

Screen handling for the blood bank is mostly done by a PC. Experience from prototypes showed that this PC client contributed significantly to the resulting response time. Three screen handling operations were selected:

transfer_screen estimates communication work when a normal simple screen image is required. A simple screen image holds only fields and rows as elements. No tables are used.

transfer_table is used when a table on the screen is read. If the user can browse through a list, this list is implemented with a table in the screen handler.

update_table accounts for the communication work when a table also can be updated.

Since these screen operations use Actor statements on the PC, `Actor_statements` are included as the fourth operation on the `screen.h` platform in Figure 7.4.

7.2.1.3 Resource Demands

Resource demands of primitive PrM processes were annotated in terms of the platform operations which were described in Section 7.2.1.2. The resource demand for one process estimates the average number of computer system resources needed per triggering of the process. Resource demands were captured during interviews with information system implementors. When every primitive process was annotated with resource demands, the implementation specification for the blood module could be derived. As a more detailed example, the resource demand for the fetch info process is shown in Figure 7.3. In Figure 7.3, it is estimated that the fetch info process uses 50 COBOL statements, one select and one transfer_screen. This resource demand reflects that this process uses a singleton select and a simple screen image.

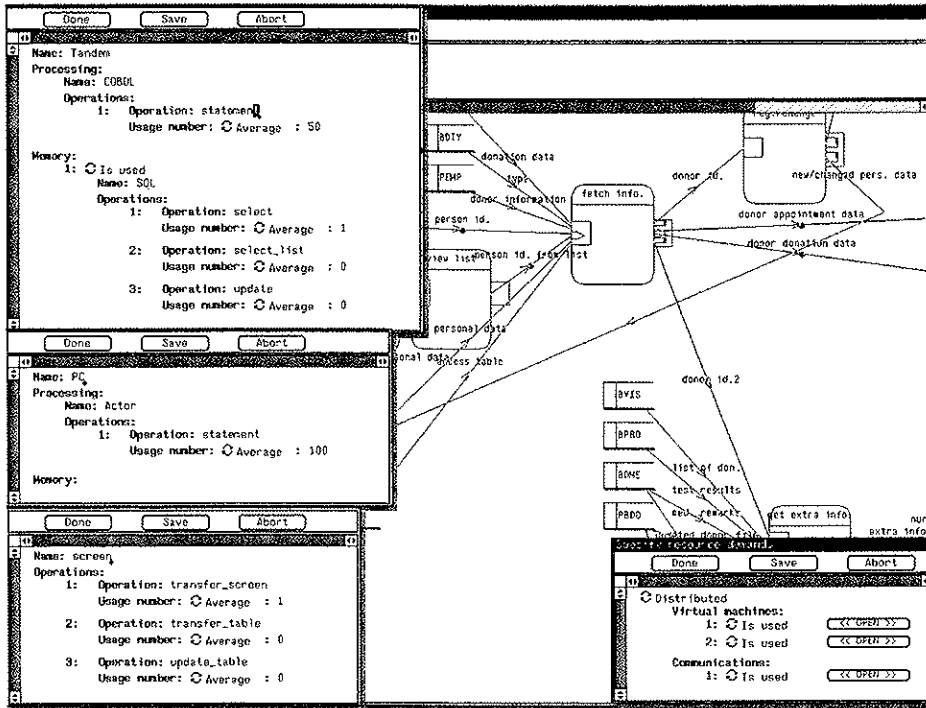


Figure 7.3: Resources used by the fetch info process.

7.3 Establish Components: System Platform

Figure 7.4 shows the SP model for the Blood Bank Case Study. The blood bank application in Figure 7.4 is termed blood. The two modules below this module in the SP hierarchy, server (or database server in full) and screen_h (screen handler

in full) constitute the platform which is used by the `blood` module. The `server` module in Figure 7.4 offers memory (database operations) and processing (COBOL-statements) to the `blood` module, while the `screen_h` module offers communication and processing to the `blood` module.

The `distr` module represents an aggregated CPU which distributes computation to one of the six physical CPUs `cpu(1) ... cpu(6)` assisted by the bus [Vet93]. The modules `diskh`, `distr`, `bus`, `cpu`, `disk`, `pc_comms`, `pc_line`, `pc_cpu`, and `distr` and all the modules used by `distr`, are also indirectly used by the `blood` module. All the other modules in Figure 7.4 are used to model the existing applications `pas`, `lab` and `mic` which are applications in the same way as the `blood` bank application.

The platform operations on the `server` and `disk_h` modules are explained below. For more information about modelling of the existing applications, consult [BOVS92, Vet93]. An outline of the hardware model can be found in [BOVS92], and is discussed further in [Vet93].

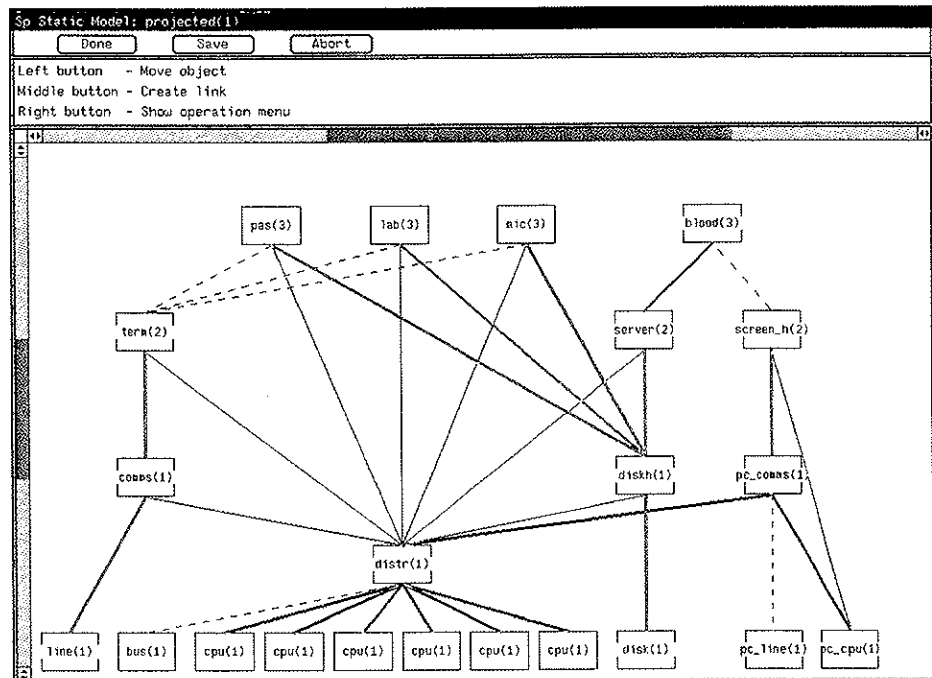


Figure 7.4: SP model for the Blood Bank Case Study

7.3.1 Database Platform Operations

Three SQL database operations are offered by the `server` module in Figure 7.4:

select This operation refers to a singleton select where one relation is referenced, and only a small volume of data returned. When the index but not the required block is stored in cache, only one block must be read from the file system for this database operation. This is indicated in the complexity specification below by the number 1. The number means that one **select** operation on the average results in one block read from the **diskh** component in Figure 7.4. In addition, a SQL message statement is used. This corresponds to the operation **sql_msg** on the **diskh** module.

select_list is used when a joined, sorted list of tuples is the result of a select. Hence, this operation induces substantially more load than the select operation. For example when the donor relation is read, in the worst case all 7000 donors are referred. One PERS tuple which stores information about one donor, occupies 260 bytes on the average. When the block size is 4 KB, maximum $\frac{7000 \cdot 260}{4 \cdot 1024} = 444$ blocks are read in the ideal case.⁸ With a selectivity of 10%, 44 of these blocks are read. This estimate is fairly representative and is therefore used in the complexity specification from the **select_list** operation to the **read_block** operation of component **diskh**. The number in the complexity function from **select_list** to **sql_msg** is 6, because each block equals 8 packets. Therefore, 44 blocks equal 6 packets.

update Consists of the three SQL operations update, insert and delete. In a stable database, these update database operations are rare, and since they require approximately the same amount of resources, there is no need to make a distinction between them. Often new tuples are added to the database even if old information is updated, making the distinction between insert and update artificial. In all three cases we assume only one write operation is used.

The matrix below represents the complexity specification for the **server** component using the **diskh** component.

$$C_{\text{diskh}}^{\text{server}} = \begin{matrix} \text{COBOL} \\ \text{select} \\ \text{select_list} \\ \text{update} \end{matrix} \begin{matrix} \text{COBOL} & \text{sql_msg} & \text{read_block} & \text{write_block} \\ \left[\begin{array}{cccc} 1 & & & \\ & 1 & 1 & \\ & 6 & 44 & \\ 1 & & 1 & 1 \end{array} \right] & & & \end{matrix}$$

For more details about this complexity specification, consult [BOVS92].⁹ The complexity specification between **diskh** and **disk** is also shown in [BOVS92].

⁸The mix of the terms "ideal" and "maximum" may be confusing. The estimate is "maximum" because all 7000 donors should be read, which corresponds to 444 blocks. On the other hand, the DBMS may read more than 444 blocks to read all the 7000 donors. In this sense, the estimate is "ideal". The estimate is therefore an "ideal maximum" or a "minimum maximum".

⁹Performance modelling of a DBMS is considered by Hyslop [Hys91]. Hyslop has designed a modelling framework with seven levels for DBMS performance modelling. This framework is supported by a tool and validated for DB2 (designed for IBM mainframes) and Teradata DBC/1012 (a multimicro database machine). Missing parameters are estimated using default values. Hyslop

7.3.2 Screen Handler Platform Operations

The `screen.h` contains processing (statements for the PC screen handler language) and communication (transfer operations). The three communication operations offered by the `screen.h` module are explained in Section 7.2.1.2. The `pc_comms` contributes with memory operations for the `screen.h` module. The `pc_comms` is distributed between the Tandem machine and the PC, and devolves both memory and processing operations on the `distr` and the `pc_cpu`. Communication between `distr` and `pc_cpu` is carried out by `pc_line`. Complexity specifications are shown in [BOVS92].

7.3.2.1 Problem with Characterisation of User Interfaces

Operations for user interfaces are in general hard to characterise, partly because there is no established way of separating requirements and design of user interfaces. In the blood bank application, the screen layout in the requirements specification was designed with the Pathmaker application in mind. Pathmaker is Tandem's mainframe screen application package. In Pathmaker, it is relatively costly to acquire each screen, therefore several fields were included in each screen image, in order to reduce the number of screens.

It was later decided to use a client-server architecture with the screen package Actor and the communication package TDS on the PC, which were communicating with the Tandem mainframe. Here, switching between windows was more efficient, and it was therefore natural to display additional information in sub-windows. Actor also provided possibilities for table lookup and selection between alternatives. Hence, the requirements specification were changed. This example reveals that, for less understood aspects of a the requirements specification, design issues are likely to blend with the requirements specification, making performance engineering harder.

7.3.3 Work Model

When combining the branching probabilities with the resource demands for each of the 13 application operations, a work complexity matrix for the blood bank application can be derived. This work model offers 13 application operations to the organisation and devolves work in terms of 8 platform operations. The resource demands for each operation in this implementation specification are the sums of the resource demands for the PrM processes which are invoked by this operation.¹⁰

circumvents the problem with the DBMS optimiser which works during run time, by actually asking the optimiser for the database in question what it would like to do.

¹⁰`tr_screen`, `tr_table` and `update_ta` in this table are abbreviations for `transfer_screen`, `transfer_table` and `update_table` respectively defined in Section 7.3.2.

	COBOL	select	select_list	update	Actor	tr_screen	tr_table	update_ta
$C_{platform}^{blood} =$	<i>search_for_donor</i>	310		1	310		1	
	<i>find_donor</i>	219	8.83		1.1	269	3.35	
	<i>specially_treated</i>	200	10		3	200		1
	<i>normal_donation</i>	400	20		9	400		1
	<i>change_prognosis</i>	400	10		2	400		1
	<i>pack_plasma</i>	400	10		4	400		1
	<i>send_plasma</i>	400	10		4	400		1
	<i>albumin_delivery</i>	400	20		4	400		1
	<i>product_delivery</i>	400	40		4	400		1
	<i>search_for_product</i>	400		1	1	400		1
	<i>fetch_patient_for_trans</i>	485	12	0.5	2	535	5.05	0.5
	<i>get_overview</i>	310	10	1	2	310	3	1
	<i>fetch_patient_for_lab</i>	210	10		2	210	3	

When update or insert database operations of vital information are done, several select operations are performed to verify that the new information is consistent with the information already stored in the database. Therefore, if one primitive PrM process consists of one update operation, often five to fifteen select operations are also included. This explains the high number of select operations, for example for the product_delivery operation.

7.4 Evaluate and Validate Static Model

Based on the work model in Section 7.3.3, and specifications of resource demands for each platform operation in [BOVS92], the time used on each operation for each resource was calculated, with the time-measure milliseconds:¹¹

	Tandem_CPU	disk	PC_line	PC_CPU	
$D_{time-work,platform}^{blood} =$	<i>search_for_donor</i>	150	88	8	820
	<i>find_donor</i>	100	37	27	1800
	<i>specially_treated</i>	130	74	30	1300
	<i>normal_donation</i>	250	200	30	1300
	<i>change_prognosis</i>	120	56	30	1300
	<i>pack_plasma</i>	140	92	30	1300
	<i>send_plasma</i>	140	92	30	1300
	<i>albumin_delivery</i>	210	110	30	1300
	<i>product_delivery</i>	360	150	30	1300
	<i>search_for_product</i>	160	110	8	820
	<i>fetch_patient_for_trans</i>	220	100	44	3100
	<i>get_overview</i>	260	140	32	2400
	<i>fetch_patient_for_lab</i>	110	56	24	1600

¹¹This matrix shows the time for each primary resource: Tandem_CPU, disk PC_line PC_CPU, and is not the direct result of matrix multiplication with the complexity matrix $C_{platform}^{blood}$. The derivation from the complexity matrix and the resource demand matrices in [BOVS92] is still straightforward.

This matrix does not include load, and is therefore not a workload model, but since it estimates residence *time* on each primary module, it is not a strict work model either. It may be termed a time-work model. From this matrix it is evident that the work on the PC is substantially higher than for the Tandem for all operations. The best-case response time for the `search_for_donor` operation is 1.1 s ($0.15 + 0.088 + 0.008 + 0.82$). This estimate does not take contention into consideration. Contention modelling is elaborated in Chapter 3.

Even with moderate contention, these response times would still be acceptable. Section 7.1.3 describes interaction with other systems and this would also affect the result. Since only a fraction of the workload affected other systems, the contribution from this will also be small. Consequently, performance objective O_1^B in Section 7.1.2 was found to be satisfactory. The performance requirement P_1^B : $X_{Blood.bank} = 100$ also seems to be satisfied when only the computerised resources were considered. As discussed in Chapter 2, the interaction between the human and computerised resources was more problematic.

When every column in the matrix for the work for each operation is multiplied by the operation arrival rate vector, a new matrix is derived. This matrix represents the workload on the Tandem and PC during the busy hour (numbers now in seconds):

$$D_{platform}^{blood} = W_{Normal}^{Blood.bank} \cdot D_{time-work,platform}^{blood}$$

$$D_{platform}^{blood} = total \begin{bmatrix} Tandem_CPU & disk & PC_line & PC_CPU \\ 9.5 & 4.2 & 2.0 & 130 \end{bmatrix}$$

During busy hour 9.5 s of Tandem CPUs, 4.2 s of Tandem disks, 2 s of PC line and 130 s of PC CPU are consumed. The result indicates that the blood bank will not constitute a significant load on the Tandem computer system during busy hour *on the average*, since 9.5 s and 4.2 s are only a small fraction of one hour. Competition for hardware resources is neglected in these estimates, as is also the case for the interaction with other systems described in Section 7.1.3. Even with these limitations it seems clear that the performance objective O_2 in Section 7.1.2 has been satisfied.

An intermediate result before the vector above was derived is shown below. This intermediate result shows how much time (in milliseconds) each operation will use on each resource during the busy hour.

	<i>Tandem_CPU</i>	<i>disk</i>	<i>PC_line</i>	<i>PC_CPU</i>
<i>search_for_donor</i>	1200	700	64	6600
<i>find_donor</i>	5900	2200	1600	106000
<i>specially_treated</i>	130	74	30	1300
<i>normal_donation</i>	500	200	60	2600
<i>change_prognosis</i>	120	56	30	1300
<i>pack_plasma</i>	280	180	60	2600
<i>send_plasma</i>	0	0	0	0
<i>albumin_delivery</i>	0	0	0	0
<i>product_delivery</i>	0	0	0	0
<i>search_for_product</i>	640	440	32	3300
<i>fetch_patient_for_trans</i>	440	200	88	6200
<i>get_overview</i>	260	140	32	2400
<i>fetch_patient_for_lab</i>	0	0	0	0

We can see that the two operations *search_for_donor* and *find_donor* account for most of the resource consumption in the blood bank application. For these two operations, the two database operations *select* and *select_list* which use Tandem CPU are the most demanding. If the model should have been refined further, only these two operations and their four resource demands need to be taken into consideration, thereby reducing the effort in parameter estimation.

7.5 Summary of Findings

For two reasons, the presentation here gives a better impression of what actually happened in the Blood Bank Case Study. First, this was a learning situation. It is therefore not considered relevant to explain errors which were done as a result of this. Second, a detailed description of the actual process would be too long and tedious to read: iterations, side steps, bottom up and top down are all used in practical modelling. As an example of iterations: an idea of the organisational operations is needed before the busy hour could be selected. Some steps may even be performed in parallel, e.g. **Estimate workload (IC.)** and **Parameterise static model components of projected application (IIA-2.)**.

In this case study, parameter validation is described in Section 7.1.4.1. Informal system validations of the blood bank applications confirm that there were no performance problems with the operations of the application, as predicted by this case study and also by the informal performance evaluation done by the project team. However, one routine in the organisation, namely the transfusion process, was problematic. This routine has been described in Section 2.1.

The primary aim of the Blood Bank Case Study was not to correct performance problems in the blood bank application, but to develop tools and methods for performance engineering during design. In this respect, this case study differed from actual performance engineering, and in this respect the case study was successful. Performance engineering would have been easier if we could augment existing conceptual models with the relevant parameters needed. In this case study, the process models had to be made from scratch, meaning an extra effort. Platform models will also be easier to make when there are performance models for the existing applications. Both these factors will make performance engineering more feasible because the cost of performance engineering will be reduced. Therefore, the scientific objective F_1^B in Section 7.1.1 was met by this case study.

The case study showed that the parameter capture for organisational data was feasible. Getting parameters for the application development was also feasible. In the Blood Bank Case Study the number of interesting numbers which must be acquired from either developers or the computer personnel are in the order of 100. Therefore, it is a manageable task to collect these parameters. The scientific objective F_2^B in Section 7.1.1 was therefore also met by this case study. Results from the case study indicate that only approximately two application operation rates and four resource demands need to be exactly specified as noted in Section 7.4. A more detailed model which also took contention into account, could investigate this more closely. This shows how the number of necessary parameters may be reduced after one iteration of parameter estimation and model prediction. In addition, this shows how personnel with low performance engineering competence but with high information system competence could help in the earlier phases. In particular, such personnel could be instrumental in resource annotation before professional performance engineers eventually take over and detail the models with the most important parameters. In this way, the scientific objective F_3^B in Section 7.1.1 is also satisfied by this case study.

Chapter 8

Organisational Change and Workflow Technology

The convergence of organisational and computational work will produce major structural changes both in organisations and in the field of computer systems, and will therefore offer new challenges for performance analysis [Den94]. An important part of the glue between the computer system and the organisation will be *workflows* which are organisational processes with a defined objective and which use both computerised and human resources. These workflows are supported by workflow systems.

This chapter forms the background for the extension of the method described in Chapter 6 to workflow systems in Chapter 9. Organisational change will first be briefly described in Section 8.1. Workflows and workflow systems are discussed in Section 8.2.

8.1 Organisational Change

The focus in this thesis is on computerised and manual *work*. For humans in an organisation, this “mechanistic” view is only one among several views. Each view offers valuable insights. This section describes the relations between a “mechanistic” view and other views. Moreover, this section introduces other valuable insights about organisational change in the context of information systems development.

Several “views” of an organisation is discussed in Section 8.1.1. Local realities, which are useful for understanding how several parts of an organisation view the organisation differently, are described in Section 8.1.2. Earlier proposals for modelling of organisations in the context of information systems are described in Section 8.1.3, with focus on OSSAD. This section concludes with a description of business process reengineering (BPR) in Section 8.1.4.

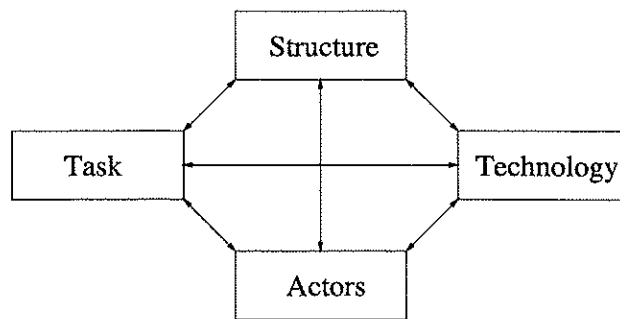


Figure 8.1: Leavitt's diamond for organisational change.

8.1.1 Ways of Viewing an Organisation

Leavitt's diamond [Lea65] in Figure 8.1 is often referred to in connection with organisational change. Leavitt views organisations as consisting of four set of variables [Lea65]:

Task: Refers to the organisations' *raison d'être*: the sequence of tasks which exist in the organisation for production of goods and services.

Actor: Describes humans with various skills.

Technology: Problem solving inventions like software, or measurement of work effectiveness.

Structure: Systems of communication, authority and work flow. The classical organisational theory which has to do with clarifying and defining the jobs of people belongs here.

Organisational change may focus on actor, technology, or structure variables, with the goal of improving organisational effectiveness. Leavitt describes how these sets of variables interact. Change in one set of variables may influence other sets of variables. Most efforts to influence organisational change must therefore deal with not only one set of variables, but eventually with all of them.

For a change process in an organisation, the actor set of variables in Leavitt's diamond are particularly important, and is the topic of several textbooks, e.g. [CH89]. A change process in organisations goes through a series of stages, each requiring considerable time [Kot95]. Skipping stages only creates the illusion of speed and may eventually stop the change process. Kotter distinguishes the eight most important phases in a change process [Kot95]: 1) establishing common understanding of urgency, 2) forming a powerful guiding coalition, 3) creating a vision, 4) communicating the vision, 5) empowering others to act on the vision, 6) planning for and creating short-term wins, 7) consolidating improvements and producing still more change, and 8) institutionalising new approaches.

Several other frameworks for understanding of organisations also exist, where Morgan presents one of the more well known [Mor88]. According to him, organisations have at least nine “images”. The mechanistic image views the organisation as an advanced machine, where formal authority, organisational charts and rules are important. The mechanic image is rational. All persons share the same goals, which is clearly a simplification. Taylor’s Scientific Management belongs to the mechanistic image. He took away the mystique from material work by breaking each task down, so that it was possible to educate unskilled workers to perfection. Drucker claims the same to be possible also for business and material work [Dru91].

Apart from the mechanistic image, Morgan describes eight other images: (1) organism, (2) brain, (3) social, (4) culture, (5) political system, (6) psychiatric prison, (7) flux and transformation and (8) instrument of domination. The cultural image for example describes myths and symbols, whereas the political system image describes competition for limited resources, conflicts, influence and power. These images are discussed in the context of workflow systems [Car95]. Aspects of these images are (of course) highly relevant for groupware systems [Gru94]. This thesis focuses on the mechanistic image.

8.1.2 Local Realities

A *local reality* represents the knowledge of individuals and groups, i.e. how these individuals or groups perceive the organisation based on their everyday experience with the organisation and from other arenas.¹ Figure 8.2 depicts how these local realities are *externalised* through communication or other actions to form the *organisational reality*, which represents how other individuals and groups have to relate to the organisation. This organisational reality may consist of institutions, language, artifacts, and technology. The organisational reality of an organisation does in turn form local realities by *internalisation* or sensemaking, which are performed more or less consciously by individuals or groups in the organisation. The resulting local reality forms the basis for how these individuals and groups act and view the actions in the organisation. Both externalisation and internalisation may happen simultaneously.

The concept of several local realities builds upon the *constructivistic* “weltanschauung”. According to the *constructivistic* “weltanschauung”, there are several realities in an organisation. This is in contrast to the *objectivistic* “weltanschauung”, where a reality exist independent of any observer, so there is only one “reality”. On the other extreme, in the *mentalist* “weltanschauung”, the concept of reality in an organisation is completely dependent of the observer. An observer can only make mental constructions of his perceptions. Therefore, the concept of a “local reality” has no meaning.

¹The theory in the beginning of this section is based on [Kro95], where a more extensive treatment and more background material can be found. Krogstie has based himself on [Gje93] and several other references which are more easily available.

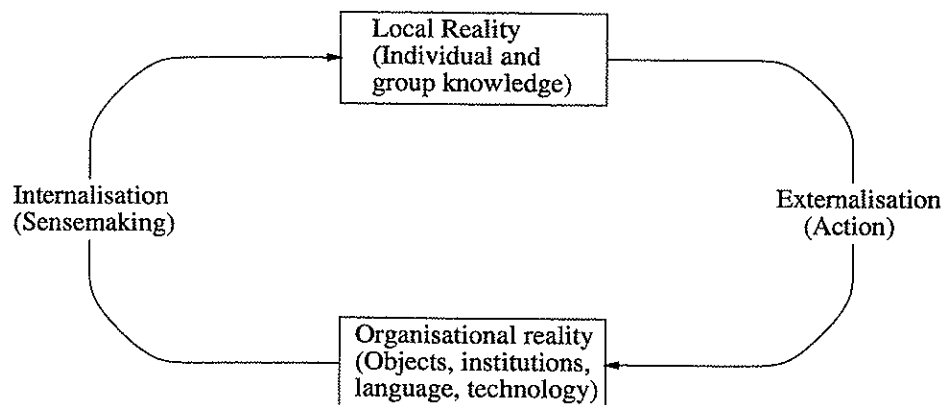


Figure 8.2: Social construction in an organisation [Gje93, Kro95].

8.1.2.1 Local Realities in Gas Sales Administration

As a small example of local realities, the reason for continuing to use telexes, which is quite slow compared to for example emails, to exchange information in the gas sales telex administration system varied through the organisation ² At least three local realities were relevant:

The data department Telex was used because Statoil, as a new partner in the gas trade, could not dictate the terms on which the trade was made. For historical reasons, telex messages are legally binding.

A contract specialist in the gas sales administration They asked the data department which arguments to use to convince their partners to change to email connections. Telex was not safer than email.

The secretary in the gas sales administration Preferred physical paper (in contrast to “virtual” paper), because when the paper is on the table, it serves as a reminder. At least the same functionality must be provided in the new system.

Externalisation of local realities is illustrated by another example from the gas sales administration system. In the Lotus Notes solution, distribution lists with a fixed number of addresses were introduced, whereas in the old manual solution the construction of a distribution was an ad-hoc activity and therefore flexible. Therefore, a local reality was externalised into an organisational reality. Even though it is possible to make ad-hoc lists in the new solution, some flexibility may be lost in the new organisational reality, and more rigidity in the organisation may result. And while the efficiency is better with the new, more routine-based solution, the effectiveness is not so easy to assess.

²This example is best understood in the context of the Gas Sales Telex Administration Case Study in Chapter 10.

8.1.3 Office Information Systems

Modelling of business processes and of information system processes will benefit from more interaction [Hol91, Ger93, Gal93]. This is the theme of office information systems. On the other hand, there must also be a separation between business models and models of the computer system, since they are not meant to solve the same problems [PW93].

Office information systems can be defined as systems which assist office workers in performing their variety of information handling tasks, such as document preparation and management, office activity planning and coordination, communication within the office and with the outside world [Yan89]. Several approaches for modelling of offices exist. OSSAD and TODOS [PBF+89, PR90] belong to the most recent research in this area. Only OSSAD is presented here, because the modelling language used in OSSAD is especially interesting. Both OSSAD and TODOS mention performance explicitly, and also reports some case studies with performance engineering. Compared to the approaches in Chapter 5, OSSAD and TODOS do not seem to offer so much new in terms of performance engineering.

OSSAD [CAS89, BC89] (Office Support System Analysis and Design) presents an integration between computerised information systems and organisational development and structure. OSSAD consists of three models:

Abstract Model models “raisons d’être” or the normative essentials of the organisation: objectives, mission, strategy. The focus is on why things are done instead of what is done. The organisation is divided into functions, which again are decomposed into sub-functions. Packets of data or objects flow between functions.

Descriptive Model concentrates on the interaction between organisational and technical solutions and contains four types of diagrams:

Role/Units Schema focuses on the interaction between roles, and is in the simplest case an organisational chart with formal authority. OSSAD makes a distinction between several roles: users, managers of organisational units being supported, developers of technical systems, developers of organisational systems. All these roles are directed towards change in the organisation, which is logical since development by definition implies change. Units in OSSAD are aggregations of roles, based on coordination/control.

Task/Procedure Schema models the relation between task/procedures and resources. Tasks are the atomic work unit performed by a single role and are aggregated into procedures with well defined input and output. Measures of performance are attached to the procedures. Resources in OSSAD are objects of data. Facilities are hardware and software.

Operations Schema is a formal description of the sequence of relationships among the operations incorporated in a task or procedure, and are mod-

elled with a Petri Net-like language. Macro-operations are defined by a role interrupt, and are generalisations of operations.

Role Interaction Schema shows the interaction between tasks and roles.

Specification Model for modelling of the software and hardware.

OSSAD uses six models which capture different aspects. This may be too complex to be of any practical use, but on the other hand, OSSAD is also a comprehensive complete modelling paradigm. Performance of an office task is for example explicitly modelled with Petri nets [Bes88].

8.1.4 Business Process Reengineering

Business Process Reengineering (BPR) is a reaction to using information technology to pave the cow paths instead of improving organisational effectiveness. BPR was originally advocated by computer scientists like Hammer [Ham90]. It is therefore not strange that the focus in the beginning was on mechanistic concepts. BPR is not based on science, but advocated by industry practitioners and consulting firms [Dav93]. It is therefore no surprise that the literature on BPR is not homogeneous [Ide95a]. In contrast to other philosophies like TQM (Total Quality Management) which aims at evolutionary changes, BPR aims at radical changes, where there is a large possibility of not reaching the goal. Both for TQM and BPR, the aim is to *plan* changes, in contrast to ad-hoc approaches. A comparison between several approaches to organisational change is shown in Figure 8.3.

According to the National Association of College and University Business Officers (NACUBO) publication, *Redesign for Higher Education*, BPR has been defined as *A managerial approach that holistically incorporates institutional strategy, work processes, people, and technology to improve effectiveness radically and to create sustainable competitive advantage by challenging and redesigning the core business processes of an institution using operational, technical, and change management in a unified way [bpr95]*. It may take some time before BPR becomes so mature.

Some writers claim BPR and workflow to be the same thing when they both are applied to the whole organisation [Ide95b]. Other writers claim BPR to be the concept for organisational change and workflow the enabling technology, where BPR precedes workflow [Mar94b]. Action Workflow described in Section 8.2.2 is one way of modelling business processes. The traditional IPO paradigm in Section 8.2.2 is another method.

8.1. Organisational Change

ELEMENT	Total Quality Management (TQM)	Just-In-Time (JIT)	Simultaneous Engineering (SE)	Time Compression Management (TCM)/ Fast Cycle Response (FCR)	Business Process Reengineering (BPR)
Focus	Quality Attitude to customers	Reduced inventory Raised throughput	Reduced time to market Increased quality	Reduced time (time = cost)	Processes Minimise non-value added
Improvement scale	Continuous Incremental	Continuous Incremental	Radical	Radical	Radical
Organisation	Common goals across functions	"Cells" and team working	R&D and Production work as a single team	Process based	Process based
Customer focus	Internal and external satisfaction	Initiator of action "pulls" production	Internal partnerships	Quick response	"Outcomes" driven
Process focus	Simplify Improve Measure to control	Workflow/Throughput efficiency	Simultaneous R&D and Production development	Eliminate time in all processes	"Ideal" or Streamlined
Techniques	Process maps Benchmarking Self-assessment SPC Diagrams	Visibility Kanban Small batches Quick set-up	Programme teams CAD/CAM	Process maps Benchmarking	Process maps Benchmarking Self-assessment IS/IT Creativity/out of box thinking

Figure 8.3: Comparing approaches to organisational change [PR95].

Davenport has devised a five step plan for BPR[DS90]:

Develop Business Vision and Process Objectives: In the most successful examples, the managers had developed a broad strategic vision where the BPR fits in. The most likely objectives are: cost, time, output quality and empowerment. It is often hard to optimise several objectives simultaneously, but focusing on one objective will often lead to improvements in the other objectives as well. The objectives should be specific and may also be quantified.

Identify Processes to Be Redesigned: Most companies are unable to support more than ten to fifteen processes per year, because of limited management attention. In the paper[DS90], two approaches are outlined for selecting processes. The exhaustive approach identifies all processes within an organisation and selects the most urgent for BPR, while the high-impact focuses on the most important processes from the start.

Understand and Measure Existing Processes: This is important so that problems are not repeated and serves as a baseline for further improvements.

Identify IT Levers: Awareness of IT capabilities should influence BPR early because IT may create new process design options. Davenport describes eight ways in which IT can influence an organisation, e.g. through disintermediation: by connection of partners who otherwise would communicate through an intermediary.

Design and Build a Prototype of a Process: BPR benefits from the iterations in prototyping. Prototyping is eased by CASE tools which generate code.

A method for BPR is also described in [GKT93].

8.1.4.1 Principles for BPR

In Hammer's article from 1990, these BPR principles are discussed [Ham90]:

Organise around outcomes, not tasks: It is better with one person than several for all subtasks: faster and closer customer contact. This principle decreases the number of subtasks, but increases the number of services each person must offer to the organisation.

Have those who use the output of the process perform the process: When the person close to the process performs it, there is little need for management. This principle shifts borders between organisations, so that the organisation which need the outputs of a given task also performs the same task, even when this normally would be another organisation's responsibility.

Subsume information-processing work into the real work that produces the information: There is no reason to separate processing and production of information. For example, in the operation ward at RiT (See Chapter 2 for a description of the blood bank case study), transfusion results were sent to the blood bank which stored them. If the operation ward could input the data, work would have been saved. This principle takes away work from other parts of the organisations.

Treat geographical resources as though they were centralised: With information systems, it is possible to have both the benefit of decentralisation and of centralisation at once. Each organisational unit does for example not need to have a purchasing department of its own. This principle reduces duplication of resources, in contrast to the second principle.

Link parallel activities instead of integrating their result using information systems for coordination during parallel activities. This principle increases coordination.

Put decision points where the work is performed and build control into the process: The people who perform the work should make the decisions.³ This principle reduces the number of authority levels by giving persons lower in the organisation more authority.

Capture information once and at the source: This principle performs storage of information once instead of several times, and has similarities with the third principle

When these principles are applied, fewer persons offer more services each. Each person also gets more responsibility. This is in contrast to scientific management, which focuses on specialisation. BPR focuses on the whole organisation and gives away control if this improves effectiveness. Interestingly, there is some similarity between these principles and Connie Smith's principles for creating responsive software in Section 5.4.1.1 [Sjø93]. This indicates that the principles are somewhat mechanistic. BPR principles could therefore be used to guide an effectiveness engineering process in the early phases of an information system development, when focus is on organisational effectiveness and not on computer system performance. Hammers's principles are similar to the four ways Drucker claims could improve the effectiveness of an organisation [Dru91]. According to Drucker, effectiveness of knowledge and service tasks could be improved in four ways [Dru91]:

1. Define the task; and eliminate unnecessary tasks. Drucker lists examples where an insurance company increased productivity of its claims-settlement department fivefold by eliminating checking on all but very large claims.
2. Concentrated work on the task; for example by using clerks to do paperwork for nurses so the nurses can concentrate on the patients. Drucker claims that

³In SP, this means moving *discrimination* (c.f. Section 4.7.2) downwards in the organisational hierarchy.

the shortage of nurses would disappear if half of their time was not spend on paperwork.

3. Define effectiveness; does it mean quantity, quality or a combination?
4. Make the employer a partner in productivity improvement; the only way to really improve effectiveness. For business processes, capital is only a *tool* in production [Dru91]. Whether tools help or harm productivity depend on the people who use them and the skills of the user. This is in contrast to the role of capital in material processes, where capital is a *factor* in production, i.e. it can replace labour.

Hammer's principles may of course be used to improve the transfusion process, for example by entering patient data directly from the ward. Actually, this was also implemented with the computerised information system, but sufficient training of the ward personnel had not been done [Sjø93]. An information system is an interaction between humans and the computerised information system, and is not stronger than the weakest link in the chain, in this case training of personnel.

8.2 Workflow and Workflow Systems

The first section gives a brief definition of workflow and workflow systems and explains the basic elements of a workflow system. Typical roles in workflows are described in Section 8.2.1.1, followed by an overview of the roots of workflow. A reference model for workflow systems is described in Section 8.2.3. A generic model of workflow system is then outlined. Section 9.3 describes the most relevant workflow taxonomy for this thesis. Two fundamentally different ways of modelling workflows are explained in the Section 8.2.2. Finally, Lotus Notes which implements some workflow features, is briefly described. Lotus Notes is relevant for this work because it was used in the Gas Sales Telex Administration Case Study.

8.2.1 What is Workflow?

A *workflow* is a *process* in an *organisation* [Mar94b, Joo95]. The workflow is assisted by a *group* of *case workers*. A typical *workflow* is shown in Figure 8.4. Figure 8.4 shows a document-based workflow which is carried out by three social resources. Each social resource is a person with some responsibilities. Petersen is the case manager who is responsible for the workflow, and Christie and Oscarson are case workers who perform some part of the workflow. In the end, there is often the need for integration of the result of each individual process. This integration is the responsibility of the case manager Peterson. This workflow was simple; most real workflows will be more complex.

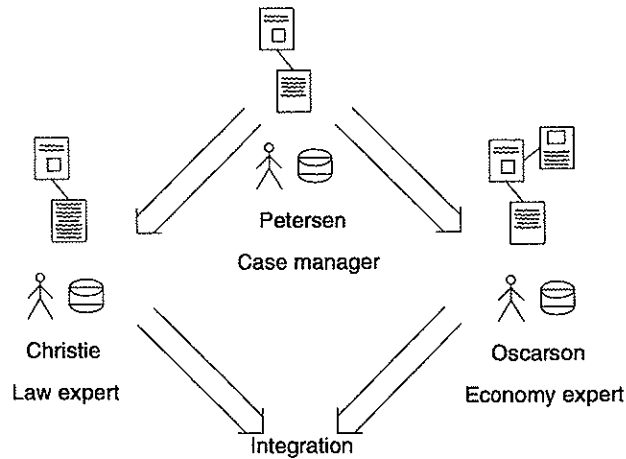


Figure 8.4: An example of a simple workflow.

During execution of a workflow, all persons involved should know where their part of the workflow fit into the overall workflow. To keep this “overview” is the task of the workflow system. A *workflow system* is a computerised information system which assists case workers who carry out workflows. A workflow system handles all the accesses to the workflow. This definition of workflow system⁴ corresponds to Marshak’s definition, who argues that workflow systems contain four basic elements [Mar94b]:

Workflows to reach business goals.

Human resources performing workflows based on business rules. By definition, workflows involve more than one person [Mar94a]. Roles which are independent of specific people are important for workflow, by increasing the flexibility.

Computerised resources invoked by the human resources for executing workflows.

Data accessed by the tools.

The same data may be used by several computerised resources. The same computerised resources may also be used by several human resources, and the same human resource may be used for several workflows. This view is also supported by the conceptual framework for workflow system and workflows found in [Joo95]. Other characteristics of workflows are:

⁴The term “task processing system” system was originally used by this author [BSH94] instead of the term “workflow system”. Since the workflow terminology is now commonly used, this non-standard terminology is no longer relevant. However, the conceptual simplicity of a *task processing system* handling a *task* supervised by a *task manager* with *task workers* carrying out *subtasks* still has appeal.

Documents: Workflow is often centred around documents which flow between the environment and the organisation and within the organisation. These documents may be electronic as well as paper-based.

Objectives: Defined before triggering the workflow and met when terminating the workflow.

Lifecycle: An organisational process requires organisation, planning, administration, supervision and execution [Sta92].

Types of processes: In an organisation there are three types of processes, namely material processes, information processes and business processes [MMWFF92]. Workflows may involve all of these processes, however, with focus on business processes.

Customers: Workflows have customers, meaning that the workflow has defined business outcome with recipients of the outcome. Customers may be either internal or external to the organisation [DS90]

Organisational boundaries: Organisational processes cross organisational boundaries, and are generally independent of formal organisational structures [DS90]. Requirements for workflow system which cross organisational borders are presented in [JLP⁺95], e.g. better monitoring tools for performance both of the workflow software and for the organisational processes running on the workflow system.

Insurance companies, banks, blood banks and governments are examples of organisations which perform workflow. Workflow systems are suited for processes where managing the flow of information is important, for example by tracking of data [CH92]. This is in contrast to information-oriented processes, where the focus is on the information content of the applications, e.g. by presenting different views to different users. Information-oriented processes may be handled by conventional information systems.

8.2.1.1 Typical Workflow Roles

Four typical roles may be identified during workflow execution [Sta92]:

Case manager has total responsibility for a workflow [Ham90]. He makes a workflow and allocates resources, ensuring completion of the workflow within the resource budget and the time limit. His task is also to modify the workflow or the resource allocation when needed.

The case manager is responsible for the workflow and may delegate work to experts where needed. The law expert and the economy expert in Figure 8.4 may work in parallel. The responsibility of the case manager is to integrate their results.

Document manager describes, registers, archives, retrieves, distributes, transforms, and throws away documents. He should ensure that documents can be reused; hence version control of documents will be an important part of his job.

Logistics manager should organise an information workflow so that subprocesses receive and send sufficient information within time limits. He is also responsible for presentation and filtering of information.

Case worker is a specialist who carries out the subprocesses [DS90] according to a workflow by getting information, establishing cooperation with other case workers, reporting status, and producing information.

Only the case worker adds value for the customer [Hal94]. The other roles are supervisory workflows where processing should be minimised as far as possible.

8.2.1.2 Taxonomies of Workflow Systems

For workflows, the taxonomy of production versus ad-hoc workflow discussed in Section 9.3 is important. For workflow *systems*, there are several taxonomies [WF94, Mar94b], e.g.:

Email-driven versus database-driven workflow: Workflow systems may be built on top of email systems or databases [Abb94]. Workflow systems on top of email systems typically occupy the low end of the workflow market, while workflow on top of databases occupies the high end of the workflow market.

Document-oriented versus process-oriented workflow: Document-oriented workflow where routing information is put into the document, e.g. intelligent documents [Yan89]. Lotus Notes is clearly document-oriented. Process-oriented workflow systems make the process explicit and are therefore better suited for more complex workflows [Abb94].

Centralised versus decentralised workflow systems: Until now, most workflow systems have a centralised workflow engine. Large organisations need a decentralised workflow system. [AMG95] proposes a decentralised architecture for the workflow system Exotica, where the workflow engine bottleneck in other workflow systems are eliminated at the expense of other problems, e.g. easy monitoring of the distributed workflows.

8.2.1.3 Related Work

In the beginning, workflow systems were developed by industry. In recent years, this topic has received academic attention. There are several roots of workflow. Each root explains one aspect of workflow:

Speech act theory aims at understanding human communication [Sea69].

Coordination science investigates coordination in different contexts. A survey of coordination science is provided in [MC94].

Groupware may be defined as a computer-based system that supports groups of people engaged in a common task (or goal) and which provides an interface to a shared environment [EGR91]. The group of people must also be aware of this sharing, in contrast to a typical transaction-oriented system which is designed to isolate users as far as possible. Workflow is only one aspect of groupware. Email is the most generally used groupware tool.

Office information systems aim at improving the understanding of organisations (see [Yan89] and Section 8.1.3.).

Project management involves project planning (i.e. defining work requirements, quantity of work and resources needed) and project monitoring (i.e. tracking progress, comparing actual to predicted, analysing impacts and making adjustments) [Ker89]. A project can be considered as any series of activities and tasks that have a specific objective to be completed within certain specifications, with defined start and end dates, funding limits (if applicable) and resources (i.e. money, people, equipment) [Ker89]. A project is usually distinguished from a production workflow by the uniqueness of the objective.

Database management system (DBMS) is the software that handles all access to the database [Dat86]. A database is a collection of stored operational data used by the application systems of some particular enterprise [Dat86]. Management of documents is an important part of workflow systems. Consult for example [Jab94, GH95] for more information about database aspects of workflow.

Operating systems is a program which acts as an interface between a user of a computer and the computer hardware. Denning argues that business processes is at the fifteenth level of abstraction in operating systems as illustrated in Figure 8.5 [Den92, Den94]. This will make operating systems “software that assists in managing the flow of work in an organisation”, in contrast to “software that manages the flow of work in a network of computers” as it is now. All the levels except form levels 14 and 15 have remained almost unchanged since the 1970s [Den92].⁵

Each of these roots has had their own view of workflow and workflow systems. Generally, the field of workflow systems is now converging. See the book [WF94] for a description of workflow and workflow systems from different angles. This book also contains five case studies and lists 63 workflow system vendors. The book [KB95] provides an introduction to both workflow, workflow systems and groupware systems. More information about requirements for future workflow systems can also be found

⁵There is a mix of processing and memory hierarchy in this figure, which may be clearer if a similar SP model had been made. The number of levels would then decrease, so level 15 would no longer exist.

	Level	Abstraction	Time scale (seconds)
multimachine levels	15	Business processes	10^5
	14	Graphical presentation of jobs	10^4
	13	User virtual machine	10^1
	12	Directories	10^{-1}
	11	Input/output streams	10^{-2}
	10	Peripheral devices	10^{-2}
	9	Files	10^{-2}
	8	Interprocess Communication	10^{-2}
singlemachine levels	7	Virtual memory	10^{-2}
	6	Local secondary storage	10^{-3}
	5	Primitive processes and semaphores	10^{-4}
	4	Interrupts	10^{-5}
	3	Procedures	10^{-6}
	2	Instruction sets	10^{-8}
	1	Local random-access memory	10^{-8}
	0	Hardware electronics	10^{-12}

Figure 8.5: Levels of abstraction in operating systems [Den92].

in [Mar92, Abb94, JLP⁺95]. A survey of workflow systems is listed in [Mar94a, JLP⁺95], and the workflow system FlowMark from IBM is described in [LA94]. Information about workflow and workflow systems can also be found at WWW, e.g. [DoCS95].

8.2.2 Workflow Modelling Approaches

Most commercial CASE tools are based on the Input/Process/Output (IPO) paradigm in Figure 8.6, for example PPP in Appendix B. Tools for modelling business processes may follow the IPO paradigm, e.g. [KLO⁺93].

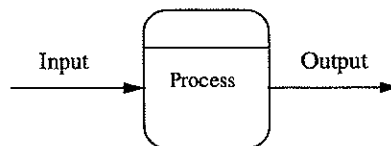


Figure 8.6: The Input/Process/Output paradigm.

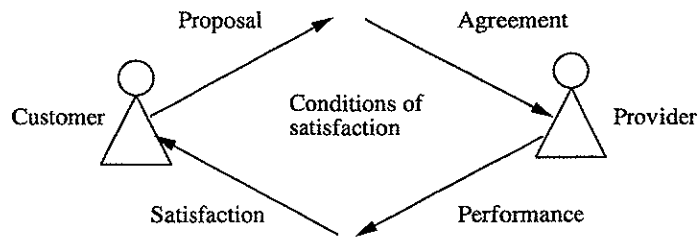


Figure 8.7: The speech act based paradigm (Adapted from [MMWFF92]. The workflow loop is copyright of Action Technology.).

Another way of modelling workflows is through the speech act based paradigm. The speech act based paradigm aims at understanding business processes and is inspired by speech act theory. The focus of the speech act based paradigm is on customer satisfaction [Red94] by explicitly modelling the interaction between the customer and the performer as a closed loop with four phases, illustrated in Figure 8.7. As described in Section 2.4, some authors view customer satisfaction as the ultimate effectiveness criteria.

During the *proposal phase*, either the customer wants some work done, and proposes conditions of satisfaction, or a performer offers some services with tentative conditions of satisfaction. The performer and the customer come to an agreement both on the workflow and on the conditions of satisfaction during the *agreement phase*. The *performance phase* finishes when the performer claims that the task has been completed. The *satisfaction phase* ends when the customer declares that the conditions of satisfaction are met. This last phase is crucial, since it ensures closing the loop [Den94]. The complete loop with all four phases has a loop time. Each phase in the loop may be visited several times before customer satisfaction is ensured. Several loops may be connected together sequentially or in parallel. There may also be conditions to be met for their execution. Extensions in queueing networks are needed to calculate contention as a result of speech act based modelling [Den94]. The interaction between SP and speech act based modelling should also be studied more carefully.

The speech act based paradigm is in some respects orthogonal to the IPO paradigm. The speech act based paradigm is good at modelling interaction between agents, where the IPO (input/process/output) paradigm is weak. On the other hand, the IPO paradigm enables hierarchic models (For a more detailed comparison, see [CH92]). An integration of the IPO paradigm and speech act based paradigm may be fruitful [Hal94]. Because each phase in the speech act based paradigm may be visited several times before customer satisfaction is reached, this integration is not straightforward.

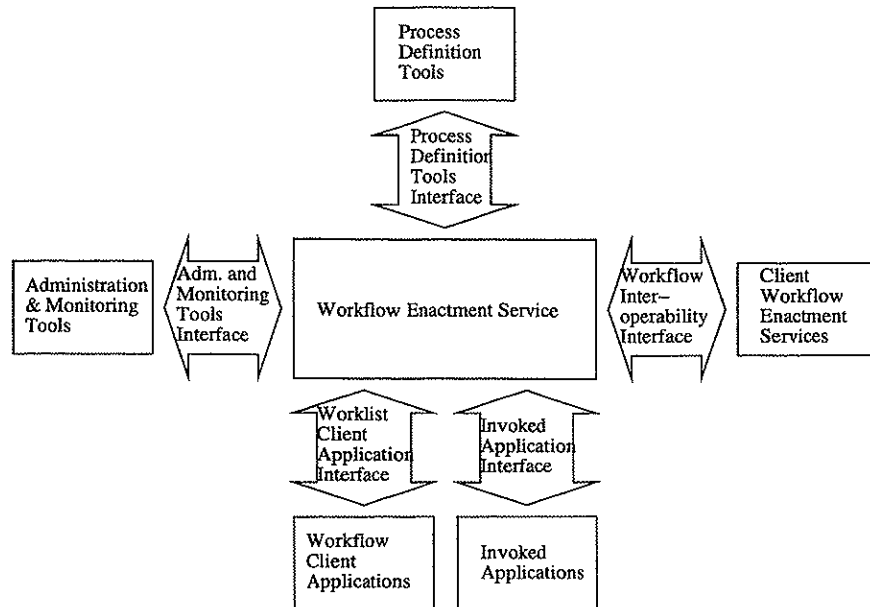


Figure 8.8: The Workflow Reference Model shows how five subsystems interoperate with the Workflow Enactment Service [Coa95].

8.2.3 Workflow Reference Model

The Workflow Coalition has designed the Workflow Reference Model depicted in Figure 8.8, to facilitate standardisation and interoperability of workflow systems. The 6 subsystems of a workflow system are described below [Coa95]:

Workflow enactment service is the basic subsystem with one or more workflow engine(s) providing the run-time environment for execution of business processes.

Process definition tools for analysing business processes. The IPO paradigm and the speech act based paradigm both explained in Section 8.2.2 are the two main ways of analysing business processes. Business Process Reengineering is intertwined with workflow (c.f. Section 8.1.4).

Workflow client applications presenting the work items to the end users. Lotus Notes is a typical example [Mar94b].

Workflow administration and management tools for presenting a view of the workflow status.

Invoked applications supplying the workflow enactment service with a range of invoked applications such as email, archive systems, and spreadsheets. Invocation of legacy systems also belongs here.

Client workflow enactment services, workflow system/engine from other vendors. Seven levels of interoperability between workflow systems are defined, ranging from the ability for a number of workflow systems to reside on the same hardware and software base with low interoperability to a similar method of workflow system operation at a high degree of interoperability.

Between the workflow enactment service and the other subsystems, interfaces enabling interoperability are defined as shown in Figure 8.8. Since interoperability is the primary objective of this reference model, these interfaces have been highlighted in the figure.

8.2.4 Lotus Notes

Lotus Notes is perhaps the best known groupware product, enabling several people to share information easily. Lotus Notes may be used as workflow client application for other workflow enactment services or process definition tools (e.g. from Action Technology). The functionality of Lotus Notes should be viewed in this context. Lotus Notes does for example not offer complete database capabilities, but uses instead other databases for advanced retrieval. Lotus Notes is not ideal for workflow either, since there is no graphical means for manipulation of the process.⁶ The simplicity of Lotus Notes is important and is assumed to be kept also in future releases.

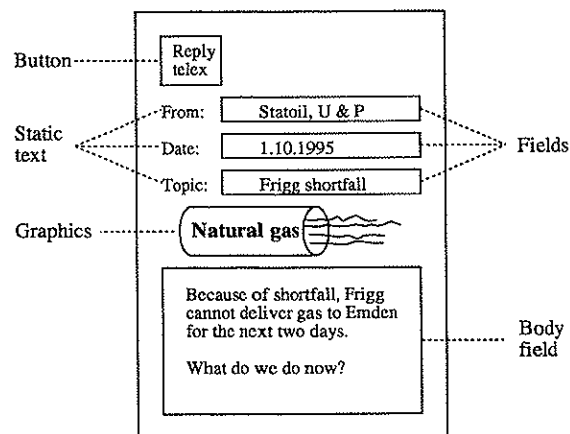


Figure 8.9: Basic elements of a form.

⁶The *workflow* module in 9.6 is almost missing in Lotus Notes which focuses on the *document* module.

30.9.1995	Total	Contract change
1.10.1995	Statoil, U & P	Frigg shortfall
1.10.1995	Statoil, M	Proposed contract annex

Figure 8.10: A view lists several fields for each form. This view is sorted on the date field.

The basic Lotus Notes functionality is simple and is outlined here. For more information about Lotus Notes, consult material from the vendor Lotus. A *database* is a collection of related *documents* stored in a single file [Lot93b]. This file may be stored in the client or in a server. Each document is defined by a *form* which specifies the content and the layout of the document. A form can contain fields, static text, graphics, and buttons as illustrated in Figure 8.9. Static text, graphics, and buttons are meant to help the user, and do not contain information which could be acted upon by other forms.

A *field* is a named area in a form which contains a single type of information, e.g. date, sender or title. The information in a field may be derived from the information in other fields. The *body field* contains free text and may in the general case be multimedia information, i.e. images and sound. In Figure 8.9 the body field contains free text. A *view* presents the information in some fields. In a view, documents are sorted based on one field. Figure 8.10 gives an example of a view where the documents are sorted on date.

The two major data structures in Lotus Notes are the *document* (which is defined in a form) and the *view*. It is illustrating to see a Lotus Notes document as a generalisation of an email message. For emails there is only one form, including the date, from, to, and subject fields. The body field is the message itself. Normally, there is only one view, which sorts the emails on date, similar to the view in Figure 8.10.

In Lotus Notes a *mailserver* is dedicated to Lotus Notes email functions and routing. A *database server* stores other non-email, shared Lotus Notes databases and replicates databases if required, but performs no email routing [Lot93a]. Replication of Lotus Notes databases in Statoil is explained in Appendix A.

8.3 Chapter Summary

The focus of this thesis is on the mechanistic aspects of organisations. Other aspects of organisations are also important for workflow. BPR is a planned approach to radical change and improved effectiveness in an organisation. Denning describes business processes as the 15th level of abstraction.

This chapter has defined workflow for the purpose of this thesis. A *workflow* is a process in an organisation using computerised and human resources and a *workflow system* which supports the execution of the workflow. Workflow has several roots

in academia, but the most important inspiration is from the market pull in industry. Since workflow is a wide area, the Workflow Reference Model is an initiative to ease the interoperability between several workflow products and an attempt to standardise the terminology in the field.

The speech act based paradigm is a new approach for specification of workflows, which may also be used as part of BPR (Business Process Reengineering). Lotus Notes is a platform for information sharing with limited workflow capabilities. Lotus Notes may also be part of a full-fledged workflow system, supplemented by other tools which interoperate according to the Workflow Management Coalition.

Chapter 9

Extending the Method to Workflow Systems

For performance engineering of workflow systems, the framework in Figure 4.3 must be extended to also take care of organisations with human resources. The basic framework was outlined in Section 2.2, and is considered in more detail in this chapter. Differences and similarities between human and computerised resources are also described. The method which was described in Chapter 6 will essentially be the same for this basic framework. The extended method is illustrated in the Gas Sales Telex Administration Case Study, especially in Chapter 10.

Section 9.1 of this chapter will first describe how SP can be used to model organisational structures. Section 9.2 describes the basic framework on this thesis. A taxonomy of the degree of determinism in workflows is described in Section 9.3. Differences between human and computerised resources are discussed in Section 9.4, before similarities between human and computerised resources are described in Section 9.5. A generic model of workflow systems is outlined in Section 9.6.

9.1 SP as Organisational Models

Inspired by [NN93], this thesis will use the following definitions: A *group* is two or more persons with common goals. More than one *group* with common goals form an *organisation*. An organisation will have at least three levels: the individual level, the group level and the organisational level as shown in Figure 9.1. Figure 9.1 is hierarchic. A large business organisation will most often have some kind of hierarchy. Hierarchy is an instrument for management and channelling of information in a complex organisation. Other organisational structures like networks are of course also possible. Typical levels in a hierarchy are the enterprise level, the business area level or the sector level as in The Gas Sales Telex Administration Case Study in Figure 10.3. Organisational resources like roles may be at a lower level of abstraction,

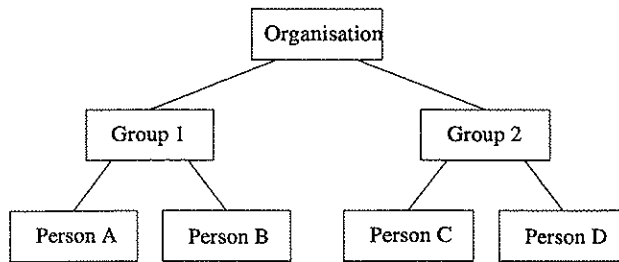


Figure 9.1: An organisation must at least consist of the individual level, the group level and the organisational level illustrated in the SP modelling style.

e.g. contract specialists and telex secretaries as seen in Figure 10.4, also in the Gas Sales Telex Administration Case Study.

9.2 Basic Framework

The basic framework for performance engineering of workflow systems was depicted in Figure 2.4 and is for convenience shown again in Figure 9.2. Since this framework subsumes the framework for performance engineering of information systems in Figure 4.3, see also Figure 4.3 for a description of the basic concepts of software and hardware. Figure 9.2 contains orthogonal concepts, e.g. workflows and resources are independent of each other in the same way as actors and processes. All concepts are shown in the same figure. The new aspects of the framework are explained below:

Workflow A workflow (process) is similar to an information system process in Figure 4.3. In addition, processes using only human resources have bold outlines.

Organisation In addition to the software/hardware hierarchies in Figure 4.3, SP will represent hierarchies in organisations as outlined in Section 9.1. All these organisational resources will in the end use operations from humans as shown in the framework in Figure 9.2.

Humans Contention modelling may be extended to human resources as well as to hardware resources. For more information on contention modelling, see Section 3.2. Extensions in queueing networks may be needed to handle the cycles described in Section 8.2.2 in workflows [Den92].

Humans in a computerised organisation will of course use software, even if this is not explicitly shown in the figure. This will be clearer in the case studies which follow.

Actors Actors start workflows and use the result of workflow. Humans are often actors. On the other hand, humans may also act as part of organisational resources which may perform work. A clear distinction cannot be made between

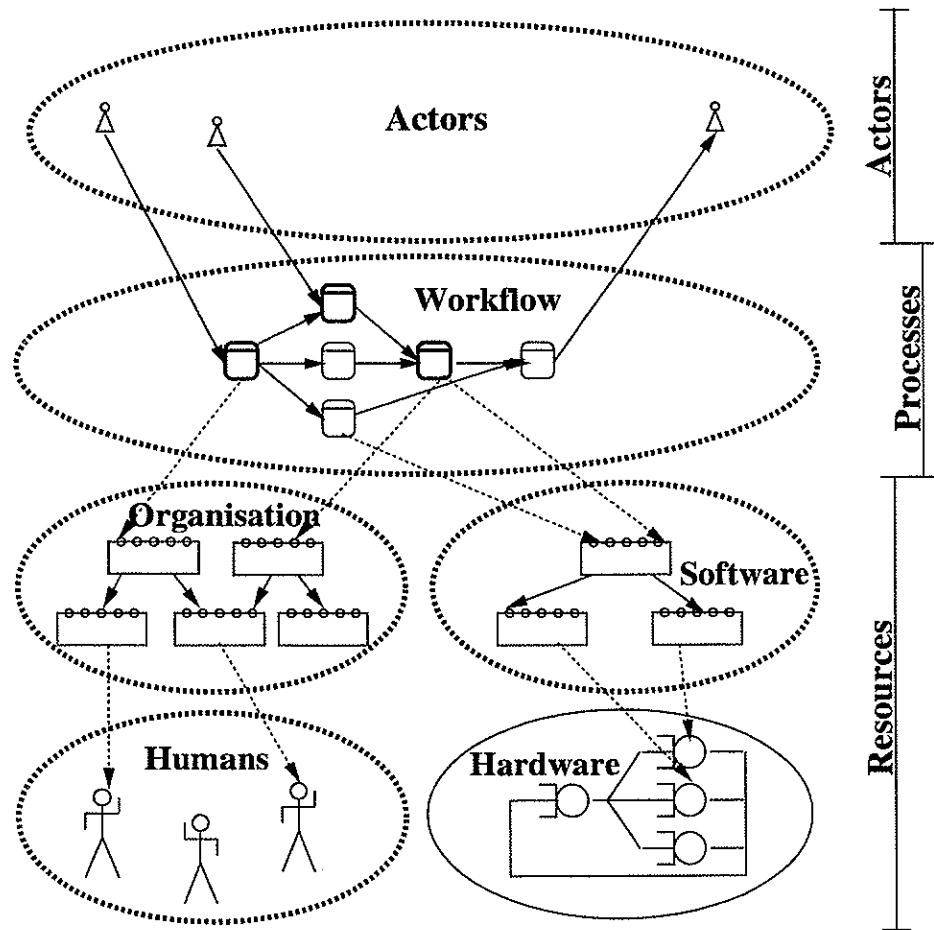


Figure 9.2: The framework for performance engineering in workflow organisations. The legend for this framework is shown in Figure 2.5 in Section 2.2.

actors and resources, because a person or a computer hardware subsystem offering operations to other actors, and therefore acting as a resource, may also become an actor when demanding work from other resources. An actor may also execute part of his workflow, and will then become a resource. Viewing a person or hardware subsystem as an *actor* or a *resource* depends on the abstraction level.

Several subsystems can be identified in Figure 2.4. Two subsystems are particularly interesting:

1. Only the organisation including human resources and the workflow which is executed in the organisation. This subsystem does not include the computerised parts of the framework, and is a purely manual information system.
2. The computerised parts in the figure, i.e. the basic software and hardware and the computerised part of the workflow, which is similar to the system in Figure 4.3.

Each of these subsystems are studied more carefully in Chapter 10 and 11 respectively in the context of the Gas Sales Telex Administration Case Study. Subsystem (1) may be used during design of the information system in order to generate performance requirements or during design of the workflow in order to derive organisational performance for this particular workflow.

It is useful to compare our framework in Figure 2.4 with Leavitt's diamond in Section 8.1.1. In our framework, parts of the task set of variables are described in the workflow system, where the sequence of the workflows is shown. The sequence of workflows is also shown in the structure set of variables. For the other three sets of variables, the mapping from Leavitt's framework to our framework is simple. The actor set of variables corresponds to our human agents, the technology set corresponds to the hardware and to the existing software in the framework. Finally, the structure set of variables corresponds mainly to the organisation in our framework. It is important to notice that the framework in this thesis does not cover all aspects of Leavitt's diamond.

9.3 Taxonomy of Degree of Determinism

Probably the most basic difference between workflow systems is the degree of determinism. This thesis uses the terms: (1) *production workflows* having deterministic resource demands, (2) *template workflows* which have statistical resource demands, e.g. some subprocesses may be optional, and (3) *ad-hoc workflows* having unpredictable resource demands, often because the next subprocess is not determined before the present subprocess is finished. From a performance point of view, a

template workflow can consist of production subprocesses, but not of ad-hoc subprocesses. If an ad-hoc workflow is part of a template workflow it is not possible to predict performance as for the template workflow. There is an interesting dynamic here: the first time a workflow is performed, it will be an ad-hoc workflow, but if the same workflow is repeated, it will then be possible to formulate template workflow or even production workflows. The three workflow types are described below:

Production workflow involving repetitive processes, critical to core business processes, e.g. insurance claims which obey a fixed sets of rules [Sil94]. In a production workflow it is possible to estimate the flow of information in advance. Therefore, the resource demands are also predefined and it is possible to minimise consumption of time and resources, with for example by approaches like BPR (c.f. Section 8.1.4). Since the workflow is predefined in detail, care should be taken *not* to deviate from the process. No workflow will be completely routine: there will always be some uncertainty.

Template workflow is a skeleton of a process where details are added during workflow execution. This template could be retrieved from an archive or may be unconsciously based on past experiences. This workflow template will tell how things are typically done; thus simplifying workflow planning. Resource demands for template workflows are stochastic. In a template workflow, the workers are empowered, i.e. by making decisions on their own (See Section 9.5 for a description of empowerment in the framework of this thesis.). The distinction between template and production workflows corresponds to the distinction between course and detailed workflow structure in [Sta92]. Workflow templates are mentioned in [LA94].

Ad-hoc workflow automates unique or occasionally used processes. Ad-hoc workflows cannot be defined in advance, often because the next subprocess is not decided before the present subprocess is finished. The resource demands for ad-hoc workflows are unpredictable, and it is therefore impossible to do management of resources here. Specialised and unstructured work belongs here, e.g. an authoring process is a design process containing some problems which are not clearly defined [TW95]. Typical for ad-hoc workflows is the need for constant communication, which after all is the very nature of “ad hoc” [KB95]. This communication may be assisted by synchronous groupware [EGR91], e.g. video conferences. Supporting ad-hoc activities with groupware may lead to large productivity gains in organisations [FRCL91]. Note that all ad-hoc work in an organisation will not be ad-hoc workflows, since a workflow to some extent may be automated.

Both the terms production workflow and ad-hoc workflow are used approximately as in the literature, e.g. [Sil94, KB95]. The concept of template workflow is made explicit here, but it is often not distinguished from a slightly more general production workflow concept. These three types of workflow represent a continuum of automated processes, not mutually distinct areas, as noted by [Mar94a] (for production and ad-hoc workflows — he did not describe template workflows.). Some workflow *systems*

will be better suited to handle production workflows, while other fit for ad-hoc workflows.

These three types of workflows are present in the case studies as follows: The blood bank workflows were production workflows which were repeated approximately in the same way several times during a month, a week, a day and even several times during an hour. In contrast, some of the processes in the gas sales administration sector in Statoil were template workflows where only the template but not the detailed workflow was repeated. This difference in determinism between four typical workflows in the two case studies is illustrated in Figure 9.3.

In the Gas Sales Telex Administration Case Study, the secretary workflow was mostly production workflow, but not as well-defined as the workflows of the Blood Bank Case Study. It should be remembered that much secretary work must be done ad-hoc. It is very hard to completely eliminate ad-hoc workflows in an organisation. Therefore, one objective of studying the production workflows is to make sure there is room for the ad-hoc workflows which are vital for the survival of the organisation. Most workflows for the contract specialists in the gas sales administration sector were of the template type. In contrast, the informal communication in Section 10.1.3.1 was of the ad-hoc type, as depicted in Figure 9.3.

The difference between the workflows of the blood bank and the sales administration sector is also reflected in the environment of the two organisations. The blood bank lived in a stable environment whereas the environment of the gas sales administration is more dynamic. The rate of change on the environment is reflected in the rate of change of the processes in the organisation, a well-known phenomena.

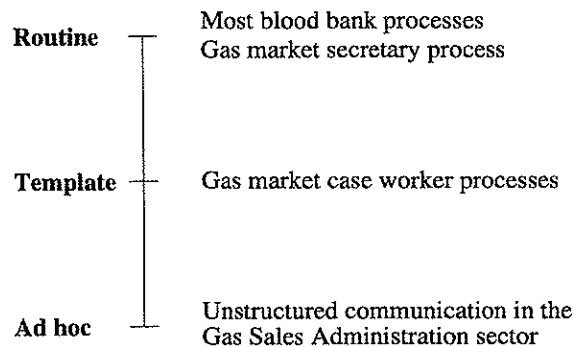


Figure 9.3: The workflows in the blood bank and in the Gas Sales Telex Administration Case Study differ in degree of determinism.

9.4 Differences between Human and Computerised Resources

The most important difference between humans and computers is that humans cover all the other “images” of Morgan outlined in Section 8.1.1, not only the mechanistic image. According to the social image, it may for example be desirable to give workers “redundant” information. Humans are irrational in addition to being rational, so it is not possible to predict their behaviour only within the mechanistic image. This is in contrast to computers which do not have judgements of their own. For computers, only the mechanistic image is relevant. Humans take initiatives (make requirements) and can perform ad-hoc work (for satisfaction of needs which are hard to formalise). Hence, it is not possible to specify a human process exactly, in contrast to a computer process where this is possible [Yu94]. There will often be a gap between the specification of a human process and the way it is actually performed. This gap should in the ideal case be non-existing for routine workflows. For a template workflow, the gap between the actual template and the specified template should also be negligible in the ideal case. For an ad-hoc workflow, it is not possible to make a specification of the workflow. When work is carried out ad hoc, the other images will also be more important than for well-defined work where constraints are fixed. A more detailed discussion would require extensive background in organisational theory and is outside of the scope of this thesis. As described in Section 10.3.2.1, human flexibility is important even in production workloads, for example if the workload is too high. Humans may handle routine work in an ad-hoc way.

Within the mechanistic image, an important difference is that the characteristic time elements will vary between human and computerised resources. The typical time element for human and computerised operations differ with several orders of magnitude. A fast CPU can perform in the order of 100 million elementary operations per second, whereas a business process may take in the order of days (10^5 seconds). This difference in characteristic time elements is also illustrated in Section 8.5. It is desirable to analyse worlds with the highest characteristic time elements before other worlds with lower characteristic time elements are analysed.

9.5 Similarities between Human and Computerised Resources

For many purposes, viewing humans and computers in the same way makes sense. This is, however, acceptable if only mechanistic aspects of humans are relevant. This is essentially done in common approaches like PERT and GANTT. Like computers, humans may also be multiprogrammed. This is described in Section 10.3.1.3, where a human can type and read at the same time. A computer may use the disk and the CPU by different processes.

As another illustration on the mechanistic view, consider the concept of workspace as defined in Section 4.6.2. A workspace of a workflow is lost when a person starts another workflow, and it will take some time before the workspace of the old workflow is restored. For a computer which stops and starts, the execution of a process with loss of workspace may result in disk usage which is expensive in terms of time. The concept of workspace in humans and machines will therefore at least have similarities, even though they may not be completely equal.

Both in organisations and in computer systems, a resource may move freely between several levels of abstraction, e.g. a human may type on the keyboard (low level), analyse the document and change it (higher), assess the benefits of the organisation and decide the scope of the document (even higher level). For computers, all the Denning's 15 abstraction levels in Section 8.2.1.3 may be used by the same CPU.

A shift in abstraction level will also give a shift in "worlds" as described in Section 6.2.2. This shift in worlds creates problems, because designers in one world will not have a complete overview of resources in other worlds. This is one of the key problems with performance engineering of information systems. In information systems, the problem is solved by giving designers in each world an overview, for example with an SP model of the total design. The world concept of information systems resembles the concept of "local realities" for organisations as elaborated in Section 8.1.2. Also for organisations, modelling may increase the overview which is needed to cope with the problems of several local realities. Several realities will also create intrinsic problems which only can be handled more or less satisfactory, however, with no "final solution".

A computerised information system and a manual information system will use two principles for shifting of work between different worlds at different levels of abstraction, namely through delegation and empowerment. Another principle for shifting work between resources at the same resource abstraction level is work switching. All three principles are depicted in Figure 9.4 and described below:

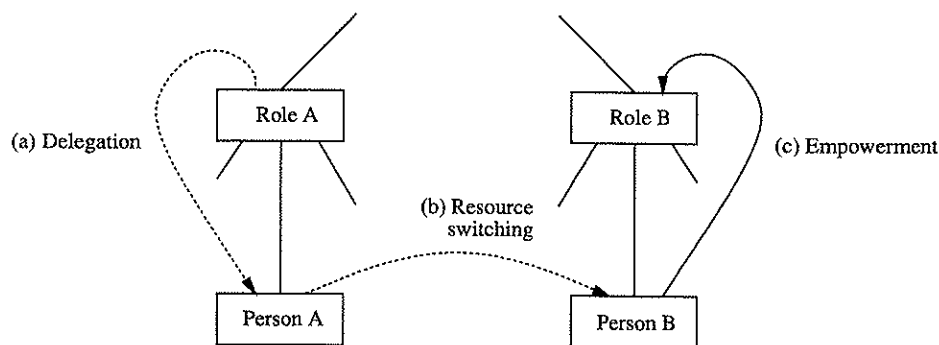


Figure 9.4: Three principles of work shifting: (a) delegation, (b) resource switching and (c) empowerment.

Delegation A top-down mechanism of shifting work between computational/organisational levels: some computational/organisational levels delegate the responsibility of some processes/workflows to a lower level in the computer/organisation. For organisations it is of course important to make sure the persons involved share the same goals.

Resource switching In this case, a resource takes care of other operations than usual. If a general resource is used to handle specialised operations, this may give inferior performance.

Empowerment In contrast to delegation which is initiated by a manager, empowerment is initiated bottom up, by a subordinate. In this case, a subordinate makes decisions which are normally made higher up in the organisational hierarchy. Often, there will be limitations as to which upper organisational level a given subordinate can operate on. Again, shared goals are crucial for the success of this principle. This principle will only apply to organisations and not to computer systems.

These three principles will interfere with each other. It is for example necessary with some sort of delegation to determine the upper organisational level for empowerment. Effective delegation will also often contain some empowerment. With empowerment, the one who delegates the work may not be aware of exactly what the person who gets the delegated responsibility for the workflow will do. The person who delegates will therefore have problems in predicting the amount of resources which will be needed.

These three principles are connected to the principles for BPR in Section 8.1.4.1 and the principles for SPE in Section 5.4.1.1. The BPR principle "Have those who use the output of the process perform the process" will for example depend on empowerment. In a computer system, resource switching is performed when a general graphics processor takes over work from a CPU, or if a dedicated document handling system takes over document handling from a database system. This is the locality-design SPE principle. Detailed evaluation of these three principles is a topic for future work.

9.6 Generic SP Model of Workflow Systems?

A workflow system offers certain operations to the organisation. An outline of a generic model of workflow systems was made in the diploma thesis of Dingstad [Jør93] under my supervision. He studied several requirements specifications for workflow systems and made a distinction between two main modules of such systems, namely the *workflow handling* and the *document handling* module (Figure 9.5). All operations in these requirement specifications manipulated either the *workflow handling* or the *document handling* module.

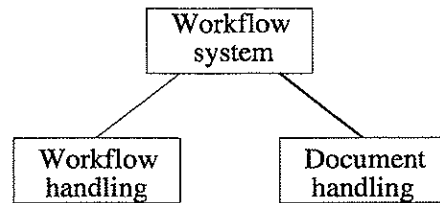


Figure 9.5: Internal architecture of a workflow system.

The *document handling* module stores documents with several versions. Typical operations are:

$$o_{\text{Document handling}} = [\text{retrieve_document}, \text{store_document}, \text{delete_document}]$$

The *workflow handling* module contains the routing workflow for the document (i.e. a process diagram) and the allocation of resources to each primary process in the workflow. Typical operations are:

$$o_{\text{Workflow handling}} = [\text{retrieve_workflow}, \text{store_workflow}, \text{delete_workflow}]$$

Process diagrams showing how the four basic roles of workflow (c.f. Section 8.2.1.1) manipulate these components are found in [Jør93]. This outline of a generic model may be extended for example by including more aspects of the Workflow Reference model in Section 8.2.3.

9.7 Chapter Summary

This chapter describes how SP may be used to model organisations. The basic framework was described in more detail in this chapter. While only the mechanistic “image” is relevant for computerised resources, all the other “images” of Morgan are relevant for humans. Therefore, it is not possible to specify human workflows or resources exactly within the mechanistic image. Within the mechanistic image, the large difference in orders of magnitude between the time element in human and computer processes and resources is important.

In this thesis, the most important taxonomy for workflows is the distinction between production workflows with deterministic resource demands, template workflows with statistical resource demands, and ad hoc workflows with unpredictable resource demands.

Chapter 10

Application to a Workflow Organisation

This chapter explains the manual processing in the Gas Sales Telex Administration Case Study. This case study was part of a project in the Norwegian oil company Statoil, termed "Better Telex in Gas Sales Telex Administration". In the gas industry, telexes are used as a means of communication between stakeholders of the gas production and trade. The central case worker in the "Better Telex in Gas Sales Administration" project was the telex secretary. The contract specialists were also important case workers. The focus of the project was improvement of telex handling effectiveness in the sales administration sector because of an anticipated increase in the number of telexes. Estimation of the performance in this case study is dependent on detailed modelling. The feasibility of this method has been explored by detailed modelling of the telex work performed by the secretary. Before selecting this case study, some requirements were defined. The case study should be:

1. Realistic: i.e. from a real organisation.
2. Extensive: i.e. include all the vital part of the proposed system. Performance is holistic, as described in Section 3.1.3. No parts of the system can be lightly ignored.
3. Illustrate design choices: The case study should illustrate *design decisions* for workflow applications. How do these decisions affect performance? Which type of decisions have to be made? On what basis can the decisions be made?
4. Small: because it must be possible to finish within the scope of this research.

All these requirements are fulfilled by this case study. The tradeoff between (2) complexity and (4) size was made easier thanks to assistance from Statoil's personnel. Since Statoil is one of the largest industrial organisations in Norway, and since Statoil is technically advanced, using state-of-the-art technology, cases from Statoil are likely

to be relevant also for other organisations. Finally, this case study continued good working relations with Statoil.

The information presented here is based on interviews with the contract specialist JRS and the secretary GVL. It is supplemented with internal reports [Wal93, Sta94b, Sta94c] where appropriate. The original Statoil project "Better Telex in Gas Sales Administration" began in March 1994, and we were involved in the project by the end of April 1994. The project "Better Telex in Gas Sales Administration" was estimated to take 600 man hours. In November 1994, time consumption was reestimated to 900 man hours and in January 1995 to 1500 man hours, because of problems with network performance in the backbone network, which takes care of the communication between internal and external networks in Statoil.

This chapter begins by providing requirements for the system in Section 10.1. The manual process for the telex secretary is parameterised in Section 10.2, and evaluated in Section 10.3. Section 10.4 describes parameters the manual process for contract specialists. The contract specialist model is evaluated in Section 10.5. Finally, this chapter ends with a summary of findings in Section 10.6.

10.1 Specify System Requirements

10.1.1 Scientific Objectives

The overall *scientific objective* of this case study is to illustrate the method for performance engineering of workflow applications, by applying the method to a real example. This will sort out the essential problems from the more academic ones.

10.1.2 Objectives of Performance Model

In [Sta94b], the objectives of the original Statoil project were stated as: (1) decrease response time (for telex handling), (2) increase throughput (for telex handling), and (3) decrease service demands (for the secretary doing telex handling). These three objectives of the original Statoil project correspond to the three main performance measures response time, throughput and utilisation as described in Section 3.1.1. Based on these three objectives of the original Statoil project, the performance objective for the performance model was formulated as:

O_1^G : Find out if response time was smaller for the computerised solution than for the manual solution.

If this performance objective was satisfied, the other two objectives of the original Statoil project were likely to be satisfied also.

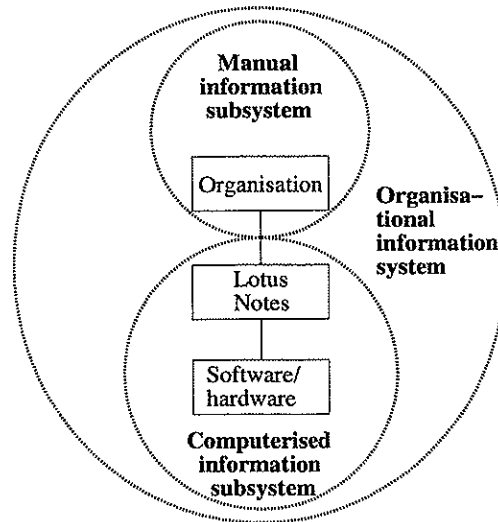


Figure 10.1: Distinction between three relevant systems

10.1.3 Determine System Boundaries

For an organisation using a workflow system, particularly two subsystems are important as described in Section 9.2. This is shown in Figure 10.1, where Lotus Notes together with associated software and hardware play the role of a simple workflow system. The computerised subsystem is described in more detail in Chapter 11. The manual information subsystem is described further below. Figure 10.1 is a simplification of Figure 9.2.¹

The rest of this section will first describe the overall process in the gas sales administration in Section 10.1.3.1, to give the context for the telex handling (sub)process. Second, the organisation in Statoil is described in Section 10.1.3.2.

10.1.3.1 Overall Model of Gas Sales Administration

An overall view on the contract coordination which is the central process for gas sales administration is given in Figure 10.2. The basic processes are:

Participate in meeting A group of companies cooperate to exploit a natural gas field in the North Sea. The relationship between these partners is regulated

¹The manual information subsystem is a subsystem if the computerised subsystem is missing, as it is in the manual solution described in this chapter. If the manual information system is put on top of the computerised subsystem it is an application and not a subsystem as defined in Section 4.3.1.

by several agreements [LGNH94]. Because of changes in the environment, e.g. production stop in the North Sea, change in cost profiles, these agreements have to be changed by annexes. These annexes are decided in formal meetings between the partners which are held approximately once a month. Change proposals from all the partners are important inputs to the meeting.

Find issues After the meetings (mentioned in process Participate in meeting), the contract coordination personnel decide on issues which need further considerations. These issues are sent to domain experts in law, economy, geology, transportation or marketing who make statements.

Make proposals for changes Out of the approximately 15 issues which are found in the **Find issues**² process, around 3 issues are critical. These critical issues are resolved based on statements from domain experts, and form the basis for change proposals.

Participate in pre-meeting The proposals for change is discussed in a pre-meeting. Also, the change proposals in these pre-meetings are sent to the other partners. The whole process then begins with a new meeting in the Participate in meeting process.

The process in Figure 10.2 is a closed process, which is a slight simplification. In addition to this formal process, there is also an informal process (informal phone calls to coordinate with partners) which is not considered here. The formal process for contract specialists has two types of (sub)processes:

Routine Consists of sending out drafts to other colleagues and finding information in the archive.

Skilled This process is harder to structure than the routine processes. Each contract specialist has some contracts to supervise. When incoming telexes arrive, contract specialists must answer questions concerning their contracts. The selected contract specialist searches in licences, agreements, committee reports and in other contracts. He also uses his own accumulated expertise, and asks other contract specialist or his manager for assistance. Since the exact wording in a telex may be vital for Statoil, more than one person must check each outgoing telex. Apart from this, there is little interaction between the contract specialists, because the large amount of details needed to handle each contract makes context changes costly.

Since the scope of the original Statoil project "Better Telex in Gas Sales Administration" was the routine processes of the contract specialist, we will also focus on these processes here. The telex secretary in the sales administration also has many processes. However, in this case study, the only relevant task is handling of incoming and outgoing telexes.

²For clarity, processes are typed with the **boldface** font.

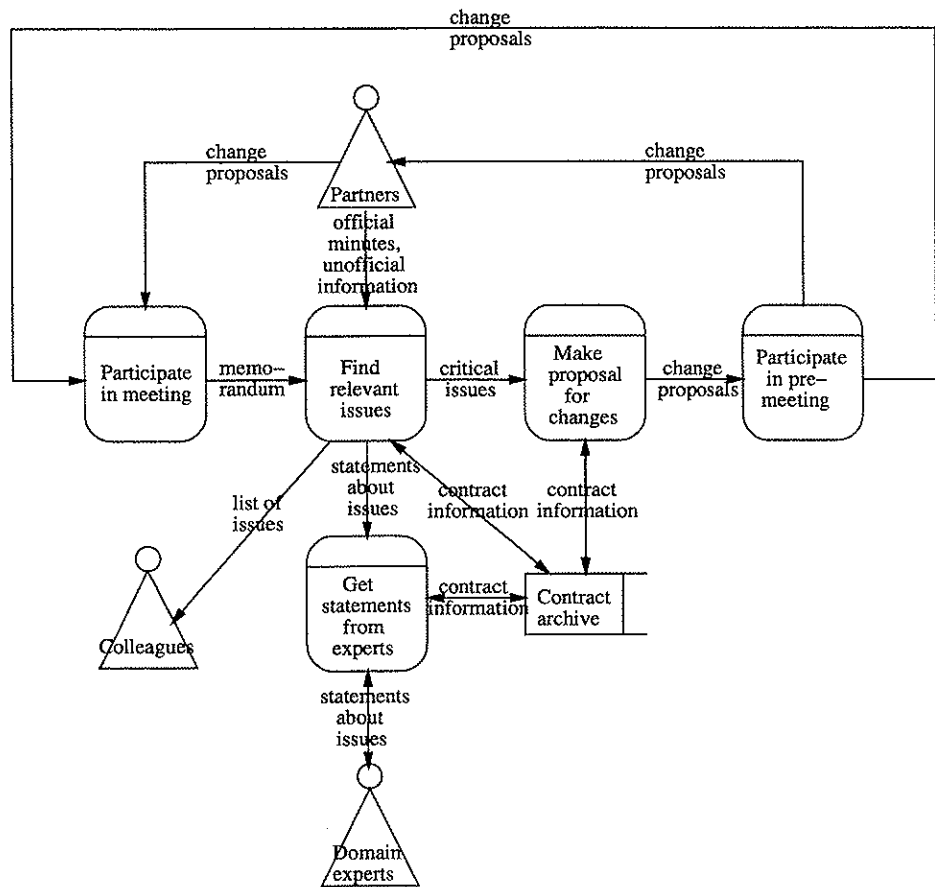


Figure 10.2: Contract coordination process.

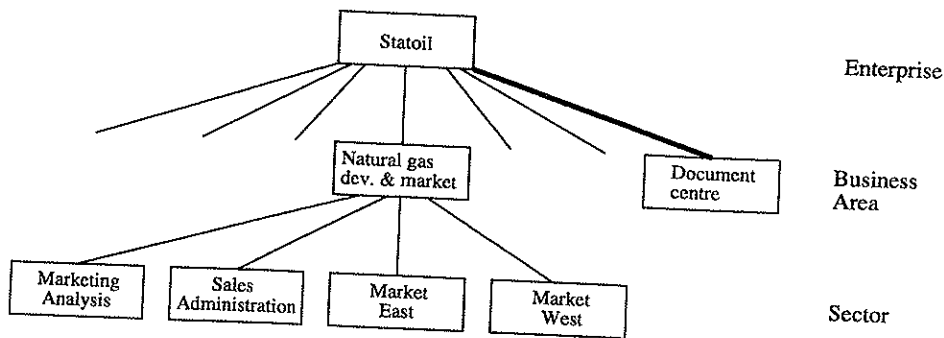


Figure 10.3: Rudimentary SP diagram for the Statoil organisation

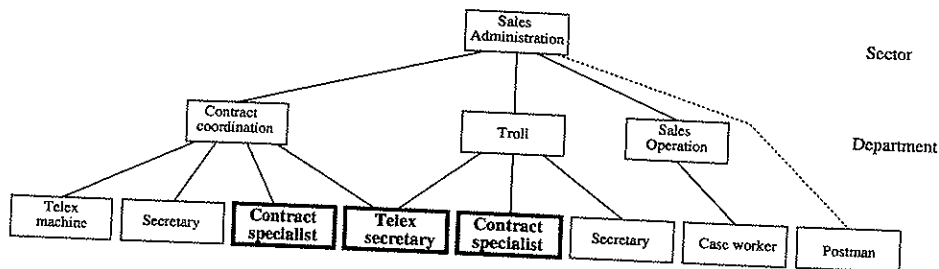


Figure 10.4: Rudimentary SP diagram for the sales administration sector.

10.1.3.2 Statoil Organisation

The organisation chart for the Natural Gas Business Area in Statoil is shown in Figure 10.3, based on [Sta94a]. Note that this organisational chart also is an SP model. Missing links show that this is not a complete SP model. As an example of memory links, the Document centre³ is shown. The Document centre will not be affected by the introduction of the "Better Telex in Gas Sales Administration" project, and is therefore not considered in detail. There are approximately 30 different Document centres in Statoil which take care of the central archiving of documents for business areas and divisions.

The organisational chart in Figure 10.3 is expanded in Figure 10.4. Figure 10.4 shows the subsystem which was discussed in Section 10.1.3.1. Both Contract coordination and Troll participate in contract coordination. Troll is a large field in the North Sea with a considerable amount of natural gas. The Sales operation department does not participate in contract coordination. The mailman is used for communication within Statoil, which is shown by the dotted line in Figure 10.4. He is not explicitly considered in this case study.

³For clarity, SP components and operations are typed with the teletype font.

The actual number of primary resources corresponding to the modelled primary resources in Figure 10.4 are in most cases 1, e.g. one telex secretary. Regarding the number of `Contract_specialists`, there are 6 both for `Contract coordination` and `Troll`. The `Sales operation` department has 30 case workers but they are not affected by telex handling. The `Mailman` may be shared by other sectors. Therefore, the average number of mailmen in the `Sales administration` will be less than 1.

From now on, the focus of the discussion will be on the resources `Telex_secretary` and `Contract_specialist` in Figure 10.4. These resources are marked with the **boldface** font in the Figure 10.4. The other secretaries in `Contract coordination` and `Troll` were not engaged in telex handling.

10.1.4 Estimate Workload

The focus of the original Statoil project “Better Telex in Gas Sales Administration” was telex handling. The rest of this thesis will focus on the `Telex_secretary` and `Contract_specialist` resources in Figure 10.4. The workload characterisation is also limited to telex handling. The name gas sales *telex* administration will be used to indicate this part of the gas sales administration which handles telexes.

Two workload scenarios are defined:

- Normal operation
- Shortfall, which happens when the seller is not able to make the contractual volumes available to the buyers, for example because of a production stop in a North Sea installation.

Contract deadline could have been defined as a third scenario, with a load between Shortfall and Normal operation. From a performance point of view, the average case (during normal operation) and the worst case (during shortfall) together give enough information.⁴

Two operations are relevant in this case study:

`in_telex` Number of incoming telexes to `Sales_telex_administration`
`out_telex` Number of outgoing telexes from `Sales_telex_administration`

If a more detailed investigation of telexes had been performed, *transactions* corresponding to meaningful organisational work, such as for example the operation

⁴In line with [Smi90], who argues that information system engineers always are optimistic, we consider best case and average estimates to be similar. Therefore, the term best case estimate will not be used in this thesis. Only average and worst case estimates are used.

Revise_contract, could have been used. Such transactions could consist of some incoming and some outgoing telexes, and could take of the order of weeks or months to terminate.

At present, normal workload per day for the gas sales telex administration is [Sta94c]:

$$W_{Today,normal}^{Sales_Telex_Administration} = \begin{bmatrix} in_telex & out_telex \\ 20 & 10 \end{bmatrix}$$

Expected future, normal workload per day is [Sta94c]:

$$W_{Future,normal}^{Sales_Telex_Administration} = \begin{bmatrix} in_telex & out_telex \\ 75 & 50 \end{bmatrix}$$

The workload during shortfall is estimated to be twice the workflow during normal operation:

$$W_{Today,shortfall}^{Sales_Telex_Administration} = \begin{bmatrix} in_telex & out_telex \\ 40 & 20 \end{bmatrix}$$

$$W_{Future,shortfall}^{Sales_Telex_Administration} = \begin{bmatrix} in_telex & out_telex \\ 150 & 100 \end{bmatrix}$$

These numbers were estimated by the contract coordinator JRS together with me.

10.1.5 Determine Performance Requirements

In Section 10.1.2, performance objective O_1^G was formulated as comparing the performance of the manual and the computerised solutions. The computerised solution should have a smaller response time than the manual solution based on the present, normal workload. This performance requirement may be expressed as:

$$P_1^G: R_{Computerised, Today, Normal}^{Sales_Telex_Administration} < R_{Manual, Today, Normal}^{Sales_Telex_Administration}$$

R is a vector of response times for each operation on the Sales Telex Administration, which in this case is the operation for incoming and for outgoing telexes.

In addition, the projected system must cope with the anticipated increase in workload from $W_{Today, Normal}^{Sales_Telex_Administration}$ to $W_{Future, Normal}^{Sales_Telex_administration}$. In line with this, a stricter performance requirement may be formulated as:

$$P_2^G: W_{Future, Normal}^{Sales_Telex_administration} \cdot R_{Computerised, Future, Normal}^{Sales_Telex_Administration} < W_{Today, Normal}^{Sales_Telex_Administration} \cdot R_{Manual, Today, Normal}^{Sales_Telex_Administration}$$

This means that the total time spent using the computerised solution under expected future workload should be less than the total time spent on the manual solution today. Note that the vector W is transposed. The product on each side of the inequality sign will be a scalar⁵.

10.2 Establish Components: Telex Secretary

A process model of incoming telexes is shown in Figure 10.5. In this figure, manual processes have bold outlines, while computerised processes have thin outlines. The processes are explained below.

Receive telex This is done by the telex machine when some contract parts have been sent by a customer or partner. The telex is brought to the mail box in the sales telex administration.

Get telex to office The telex must be brought from the mail box to the telex secretary's office.

Read telex The telex secretary must read the telex to determine who she should send it to.

Log telex For each telex information about the sender, the topic and other information must be filed in a log.

⁵For P_1^G , the workload does not have to be taken into consideration, since the workload is similar on both sides of the inequality sign. The inequality sign in P_1^G will compare two vectors, in contrast to P_2^G where two scalars are compared. This difference could have been eliminated if P_1^G had been formulated as:

$$P_2^G: W_{Today, Normal}^{Sales_Telex_administration} \cdot R_{Computerised, Today, Normal}^{Sales_Telex_Administration} < W_{Today, Normal}^{Sales_Telex_Administration} \cdot R_{Manual, Today, Normal}^{Sales_Telex_Administration}$$

This more complex formulation of P_1^G is *almost* similar to the simpler formulation which is used in this thesis. The extra complexity with the more complex formulation was not considered necessary.

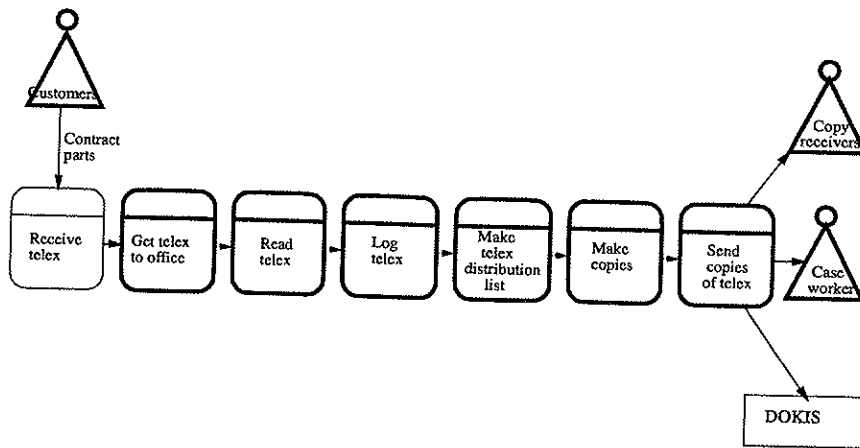


Figure 10.5: Process for incoming telexes.

Make telex distribution lists The secretary makes a list of who should receive each telex.

Make copies She copies the telex so that each receiver gets a copy of the telex.

Send copies of telex The telex secretary sends copies of the telex to copy receivers and case workers internal to the sales administration in Statoil according to the distribution list which she made earlier.

The process for outgoing telexes are shown in Figure 10.6 and the processes are explained below.

Transport from case worker Each telex is brought from the case worker to the telex secretary.

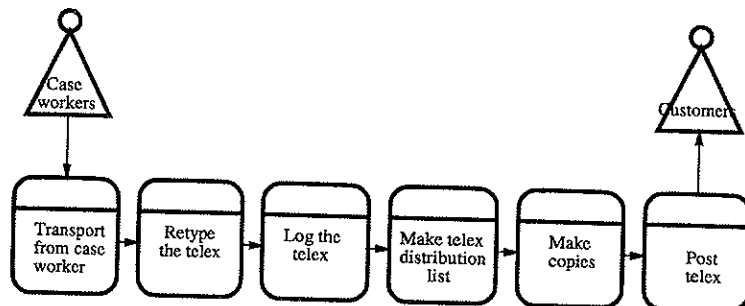


Figure 10.6: Process for outgoing telexes.

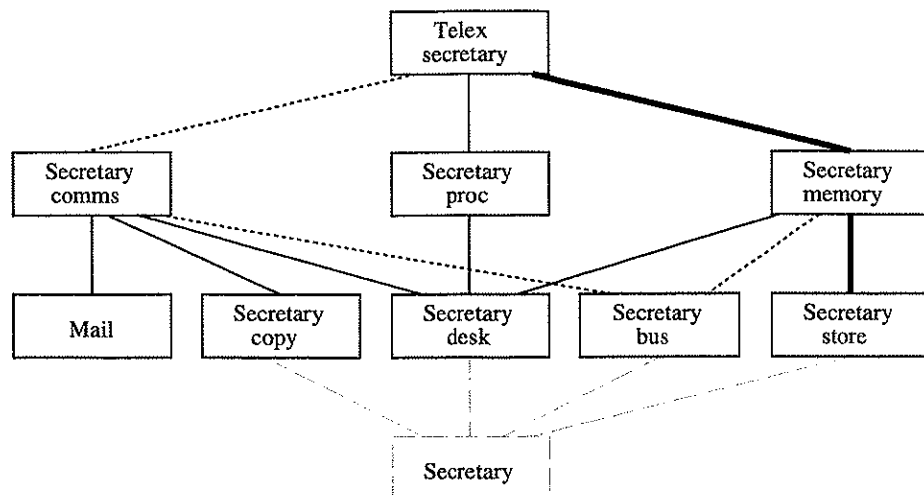


Figure 10.7: SP model for the present telex handling by the telex secretary.

Retype the telex The telex secretary retypes the telex text, because the computer system of the contract specialists is not integrated in the same network as used to send telexes.

Log the telex Each outgoing telex is logged for future reference.

Make telex distribution list The secretary makes a list of who should receive each telex.

Make copies She copies the telex so that each receiver gets a copy of the telex.

Mail telex The telex secretary mails copies of the telex to receivers external to the sales telex administration according to the distribution list which she made earlier.

Mail telex The telex secretary mails copies of the telex to receivers external to the sales telex administration according to the distribution list which she made earlier.

Based on the process diagrams above, the SP model in Figure 10.7 is derived. The resource `Telex_secretary` is visible to the outside world. `Mail` is also used by several other resources. All the other resources in Figure 10.7 are only visible to the `Telex_secretary`. They are in `Telex_secretary`'s world. Internally, `Telex_secretary` plays *roles* in communication, processing and memory. This is indicated by placing `Secretary_comms`, `Secretary_proc` and `Secretary_memory` under `Telex_secretary` in the figure. On the bottom level, `Secretary_copy`, `Secretary_desk`, `Secretary_bus`, and `Secretary_store` perform the basic operations of

the telex secretary.⁶ The external resource `Mail` is also used by `Secretary_comms`. Each component and complexity specification is described in detail in the next subsections.

As explained in Section 10.1.4, `Sales Telex Administration` has the two operations `in_telex` and `out_telex`. Part of the work in these two operations is delegated to `Telex_secretary`. Therefore, `Telex_secretary` also has the two operations `in_telex` and `out_telex`.

10.2.1 Telex Secretary's Role in Communication

A telex is distributed to external and internal receivers, i.e. receivers outside and inside of the sales telex administration in Statoil. In Figure 10.7, `Secretary_comms` accounts for the communication between the telex secretary and the internal and external receivers of telexes. The link from `Telex_secretary` to `Secretary_comms` in Figure 10.7 is a communication link, because `Telex_secretary` delegates communication to `Secretary_comms`.

Figure 10.5 shows the process for incoming telexes. `Telex_secretary` has to perform the process **Get telex to office** and afterwards the process **Send copies of telex** (to the internal receivers) in this figure. The other processes in Figure 10.5 do not involve communication by `Telex_secretary`. These two communication processes result in the two operations `fetch_mail` and `send_internal` telex as shown in the complexity specification below.

$$C_{\text{Secretary_comms}}^{\text{Telex_secretary}} = \begin{matrix} \text{in_telex} \\ \text{out_telex} \end{matrix} \begin{matrix} \text{send_draft} & \text{send_internal} & \text{fetch_mail} & \text{send_mail} & \text{fetch_receipt} \\ \left[\begin{array}{ccccc} 0 & 1 & 1 & 0 & 0 \\ x & 1 & 0 & 1 & 1 \end{array} \right] \end{matrix}$$

The parameter x in this complexity specification receives a value in Table 10.1. For outgoing telexes, the telex secretary must do the process **Retype the telex**. In this retyping, there are x drafts of the telex distributed to the case worker who prepares the draft. This is the reason for the x `send_draft` operations for each outgoing telex (the `out_telex` operation). Moreover, `Telex_secretary` has to `send_internal` telexes and `send_mail` to the telex office via the mail box. For outgoing telexes there will also be a receipt where there is a need for communication.

⁶In Figure 10.7, the components `Secretary_copy`, `Secretary_desk`, `Secretary_bus`, and `Secretary_store` are implemented by the same person, and are therefore not four independent subsystems as described in Section 4.3.1. The common, *primary* subsystem consisting of the `Secretary`, is depicted with the dashed box in the figure. The dashed box is not relevant for static modelling, only for dynamic modelling. Therefore, the dashed box with corresponding complexity specifications is not relevant in this chapter, which focuses primarily on static modelling.

10.2.1.1 Copying of telexes

The link from the `Secretary_comms` module to the `Secretary_copy` module is a processing link. For the `send_internal` (copy of telex) operation, one copy of n_p pages in the telex is needed for each of the r_i internal receivers. For the other operations, no copying is needed.

$$C_{Secretary_copy}^{Secretary_comms} = \begin{array}{l} \textit{send_draft} \\ \textit{send_internal} \\ \textit{fetch_mail} \\ \textit{send_mail} \\ \textit{fetch_receipt} \end{array} \begin{array}{c} \textit{copy} \\ \left[\begin{array}{c} 0 \\ n_p r_i \\ 0 \\ 0 \\ 0 \end{array} \right] \end{array}$$

10.2.1.2 Desk processing

In addition to `Secretary_copy`, `Secretary_comms` also delegates processing to `Secretary_desk`. For the `fetch_mail` and `fetch_receipt` operations, a split of the telex is needed. Both `fetch_mail` and `send_mail` are stamped with logging information like date, who the receipts of the telex are, etc. The receipt must be compared with the n_t telexes with n_p pages each. Receipt handling is therefore modelled as reading n_t telexes with n_p pages each.

$$C_{Secretary_desk}^{Secretary_comms} = \begin{array}{l} \textit{send_draft} \\ \textit{send_internal} \\ \textit{fetch_mail} \\ \textit{send_mail} \\ \textit{fetch_receipt} \end{array} \begin{array}{c} \textit{split} \textit{stamp} \textit{read} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & n_t n_p \end{array} \right] \end{array}$$

10.2.1.3 Mail

`Secretary_comms` uses the mail box as storage for the mail which is sent and received. This link is not a communication link, because the message which is sent in the mail box is more persistent than `Secretary_comms`. The complexity specification is shown below:

$$C_{Mail}^{Secretary_comms} = \begin{matrix} send_draft \\ send_internal \\ fetch_mail \\ send_mail \\ fetch_receipt \end{matrix} \begin{matrix} send_mail & get_mail \\ \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{array} \right] \end{matrix}$$

10.2.1.4 Walking

The walking (which has the *communication* operation type, as shown by the links in Figure 10.7) for the *Secretary_comms* is handled by *Secretary_bus*. The walking distances to the mail box, copier and contract specialist are d_p , d_c and d_s , respectively. For example a *send_draft* operation needs walking to the contract specialists.

$$C_{Secretary_bus}^{Secretary_comms} = \begin{matrix} send_draft \\ send_internal \\ fetch_mail \\ send_mail \\ fetch_receipt \end{matrix} \begin{matrix} walk \\ \left[\begin{array}{c} d_s \\ d_c + d_p \\ d_p \\ d_p \\ d_p \end{array} \right] \end{matrix}$$

10.2.2 Telex Secretary's Role in Storage

Secretary_memory in Figure 10.7 is the secretary acting as storage and therefore has a memory link. The secretary stores addresses of internal and external telex receivers and also logs the telexes. Moreover, telexes are stored in the local archive. This is represented in the complexity specification below:

$$C_{Secretary_memory}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{matrix} get_int_adr & get_ext_adr & store_arch & get_arch & store_log & get_log \\ \left[\begin{array}{cccccc} r_i + 1 & 0 & 1 & 0 & 1 & 0 \\ r_i & r_c & 1 & 1 & 1 & 1 \end{array} \right] \end{matrix}$$

10.2.2.1 Processing

For each of the operations in *Secretary_memory*, some reading must be performed. It is estimated that approximately 0.1 page are read for each operation:

$$C_{\text{Secretary_memory}}^{C_{\text{Secretary_desk}}} = \begin{array}{l} \text{get_int_adr} \\ \text{get_ext_adr} \\ \text{store_arch} \\ \text{get_arch} \\ \text{store_log} \\ \text{get_log} \end{array} \begin{array}{l} \text{read} \\ \left[\begin{array}{l} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{array} \right] \end{array}$$

The link from `Secretary_memory` to `Secretary_desk` is a processing link.

10.2.2.2 Walking

When the local archive is used, the secretary has to walk back and forth between her desk and the local archive.

$$C_{\text{Secretary_memory}}^{C_{\text{Secretary_bus}}} = \begin{array}{l} \text{get_int_adr} \\ \text{get_ext_adr} \\ \text{store_arch} \\ \text{get_arch} \\ \text{store_log} \\ \text{get_log} \end{array} \begin{array}{l} \text{walk} \\ \left[\begin{array}{l} 0 \\ 0 \\ d_a \\ d_a \\ 0 \\ 0 \end{array} \right] \end{array}$$

This complexity specification is a communications link as shown in Figure 10.7.

10.2.2.3 Interface to Storage

The telex secretary acts as an interface to the archive and to the log and the address list for external and internal addresses. The work of getting data to and from the memory is shown by the memory link. Each operation in `Secretary_memory` is mapped directly to a primitive operation in `Secretary_store`.

$$C_{\text{Secretary_memory}}^{C_{\text{Secretary_store}}} = \begin{array}{l} \text{get_int_adr} \\ \text{get_ext_adr} \\ \text{store_arch} \\ \text{get_arch} \\ \text{store_log} \\ \text{get_log} \end{array} \begin{array}{l} \text{get_int_adr} \quad \text{get_ext_adr} \quad \text{store_arch} \quad \text{get_arch} \quad \text{store_log} \quad \text{get_log} \\ \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right] \end{array}$$

This complexity specification is the identity matrix:

$$C_{Secretary_store}^{Secretary_memory} = I$$

10.2.3 Telex Secretary's Role as Processor

For outgoing telexes, `Telex_secretary` has to retype n_p pages written by the contract specialist into the text processing system Display Write 370 [Wal93]. There are also x iterations between the contract specialist and the secretary with draft versions of outgoing telexes with n_p pages which the `Telex_secretary` has to polish:

$$C_{Secretary_proc}^{Telex_secretary} = \begin{array}{c} in_telex \\ out_telex \end{array} \begin{array}{c} retype \quad polish \\ \left[\begin{array}{cc} 0 & 0 \\ n_p & xn_p \end{array} \right] \end{array}$$

Both for retyping and polishing one page, reading one page is necessary. Whereas retyping uses text which has been used in other documents, polishing uses text in the same draft as a basis. In both cases, only a fraction of the text is typed again. The fraction of pages typed again will differ between retyping (p_s) and polishing (p_i):

$$C_{Secretary_desk}^{Secretary_proc} = \begin{array}{c} retype \\ polish \end{array} \begin{array}{c} read \quad type \\ \left[\begin{array}{cc} 1 & p_s \\ 1 & p_i \end{array} \right] \end{array}$$

10.2.4 Total Secretary Work

This section presents the total work which `Telex_secretary` delegates to the primitive resources `Mail`, `Secretary_copy`, `Secretary_desk`, `Secretary_bus`, and `Secretary_store`.

10.2.4.1 Mail

The total work on the `Mail` resource is:

$$C_{Mail}^{Telex_secretary} = C_{Secretary_comms}^{Telex_secretary} \cdot C_{Mail}^{Secretary_comms}$$

$$C_{Mail}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{matrix} send_mail & get_mail \\ \left[\begin{array}{cc} 0 & 1 \\ 1 & 1 \end{array} \right] \end{matrix}$$

10.2.4.2 Secretary_copy

For Secretary_copy, the total work is:

$$C_{Secretary_copy}^{Telex_secretary} = C_{Secretary_comms}^{Telex_secretary} \cdot C_{Secretary_copy}^{Secretary_comms}$$

$$C_{Secretary_copy}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{matrix} copy \\ \left[\begin{array}{c} n_p r_i \\ n_p r_i \end{array} \right] \end{matrix}$$

10.2.4.3 Secretary_desk

Total work on Secretary_desk:

$$C_{Secretary_desk}^{Telex_secretary} = C_{Secretary_comms}^{Telex_secretary} \cdot C_{Secretary_desk}^{Secretary_comms} + C_{Secretary_memory}^{Telex_secretary} \cdot C_{Secretary_desk}^{Secretary_memory} + C_{Secretary_proc}^{Telex_secretary} \cdot C_{Secretary_desk}^{Secretary_proc}$$

$$C_{Secretary_desk}^{Telex_Secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{matrix} split & stamp & & read & & type \\ \left[\begin{array}{cccccc} 1 & 1 & & 0.1(r_i + 3) & & 0 \\ 1 & 1 & (1 + x + n_t)n_p + 0.1(4 + r_i + r_e) & & n_p(p_s + xp_i) & \end{array} \right] \end{matrix}$$

10.2.4.4 Secretary_bus

Delegated work on Secretary_bus:

$$C_{Secretary_bus}^{Telex_secretary} = C_{Secretary_comms}^{Telex_secretary} \cdot C_{Secretary_bus}^{Secretary_comms} + C_{Secretary_memory}^{Telex_secretary} \cdot C_{Secretary_bus}^{Secretary_memory}$$

$$C_{Secretary_bus}^{Telex_Secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{matrix} \\ walk \\ \left[\begin{array}{c} d_a + d_c + 2d_p \\ 2d_a + d_c + 3d_p + xd_s \end{array} \right] \end{matrix}$$

10.2.4.5 Secretary_store

On Secretary_store, the total work is:

$$C_{Secretary_store}^{Telex_secretary} = C_{Secretary_memory}^{Telex_secretary} \cdot C_{Secretary_store}^{Secretary_memory}$$

$$C_{Secretary_store}^{Telex_Secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{matrix} \\ \\ \left[\begin{array}{cccccc} get_int_adr & get_ext_adr & store_arch & get_arch & store_log & get_log \\ r_i + 1 & 0 & 1 & 0 & 1 & 0 \\ r_i & r_e & 1 & 1 & 1 & 1 \end{array} \right] \end{matrix}$$

10.3 Evaluate and Validate Static Model of Telex Secretary

In this section, the work complexity functions for the telex secretary are used to estimate the performance of the telex secretary. Section 10.3.1 estimates resource demands on the primary telex secretary operations, and in Section 10.3.2 the performance for the telex secretary is estimated.

10.3.1 Resource Demands for Telex Secretary

The resource demands for all the primary operations are estimated in this section. It is necessary to replace parameter names by numbers. Table 10.1 below shows the average numbers which are used. As an example, the average number of external receivers $r_e = 9$ in the table is estimated as follows: There are from 4 to 8 firms with

Parameter	Value	Description of parameter
n_t	2	Number of telexes to be read during receipt handling
n_p	2	Number of pages in a telex, from 1 to 5.
d_a	1	Walking distance, in meters, to local archive.
d_c	10	Walking distance, in meters, to copier.
d_p	2	Walking distance, in meters, to the mail box.
d_s	10	Walking distance, in meters, between contract specialist and telex secretary
r_e	9	External receivers of a telex.
r_i	2	Internal receivers of a telex.
x	1	Number of iterations between contract specialist and telex secretary
p_i	0.5	Fraction of pages which has to be retyped initially by the telex secretary based on text from the contract specialist.
p_s	0.25	Fraction of pages to be retyped during each iteration between between contract specialist and telex secretary

Table 10.1: Parameters representing average numbers for the manual solution.

1 to 2 receivers in each firm. In addition, there are 2 receivers working in Statoil, but not in Stavanger (the “oil capital of Norway”, where the sales telex administration is located), who will be considered as external to the sales telex administration. In the rest of this section, these numbers are used to calculate numeric estimates for resource demands for the telex secretary.

10.3.1.1 Resource Demands for Mail Handling

The two parameters k_s and k_g are used for resource demands of mail handling, giving the resource demand matrix as follows:

$$D_{Mail} = \begin{matrix} send_mail \\ get_mail \end{matrix} \begin{bmatrix} k_s \\ k_g \end{bmatrix}$$

The total resource demand for mail handling is:

$$D_{Mail}^{Telex_secretary} = C_{Mail}^{Telex_secretary} \cdot D_{Mail}$$

$$D_{Mail}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} k_g \\ k_s + k_g \end{bmatrix}$$

10.3.1.2 Resource Demands for Secretary as Copy Operator

The `Telex_secretary` must operate the copier. This is estimated to take 0.02 minutes per copy, and the resource demand matrix therefore becomes:

$$D_{Secretary_copy} = copy \begin{bmatrix} 0.02 \end{bmatrix}$$

Copying of telexes is cumbersome, because of the thin paper which is produced by the telex machine. The total resource demand for the secretary as copy operator is:

$$D_{Secretary_copy}^{Telex_secretary} = C_{Secretary_copy}^{Telex_secretary} \cdot D_{Secretary_copy}$$

$$D_{Secretary_copy}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 0.02 \cdot (2 \cdot 2) \\ 0.02 \cdot (2 \cdot 2) \end{bmatrix}$$

$$D_{Secretary_copy}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 0.08 \\ 0.08 \end{bmatrix}$$

10.3.1.3 Resource Demands for Secretary as Desk

As a processor, the telex secretary offers the operations `split`, `stamp`, `read` and `type`. The “speed” of these operations is estimated below. To `split` a telex takes 0.25 minutes, and to `stamp` a telex takes 1 minute. The measure for `read` is the number of pages to read. One page takes 1 minute, so the time spent for reading is linear with the number of pages. Typing one page is estimated to take 10 minutes.

$$D_{Secretary_desk} = \begin{matrix} split \\ stamp \\ read \\ type \end{matrix} \begin{bmatrix} 0.25 \\ 1 \\ 1 \\ 10 \end{bmatrix}$$

The relationship between the number of pages read and the time it takes need not be linear, i.e. for less than one page, reading is slower than for more than one page, except when the number of pages reaches a certain limit, when reduced attention reduces the speed again. Another solution would be to use time to read as a measure, but if reading speed is not equal, this will clutter the separation between work and load. This also applies to the operation type.

Typing could be combined with reading, just as the secretary could think while she walks. Thus, a human agent could perform several functions at a time, each using different aspects of the human agent (which is analogous to multiprogramming in a computer system, i.e. the disk and the CPU may be used by different processes). The modelling language allows for all this complexity to be represented.

The total resource demand for the secretary as desk is:

$$D_{Secretary_desk}^{Telex_secretary} = C_{Secretary_desk}^{Telex_secretary} \cdot D_{Secretary_desk}$$

$$D_{Secretary_desk}^{Telex_secretary} = \frac{in_telex}{out_telex} \left[\begin{array}{c} 0.25 \cdot 1 + 1 \cdot 1 + 1 \cdot 0.1(2 + 3) + 0 \\ 0.25 + 1 + [(1 + 1 + 2) \cdot 2 + 0.1(4 + 2 + 9)] + 10[2(0.25 + 1 \cdot 0.5)] \end{array} \right]$$

$$D_{Secretary_desk}^{Telex_secretary} = \frac{in_telex}{out_telex} \left[\begin{array}{c} 1.75 \\ 25.75 \end{array} \right]$$

10.3.1.4 Resource Demands for Secretary as Bus

When the secretary works as a bus, she walks. The "speed" of the secretary is estimated to be 0.025 minutes per meter. The walking "speed" is valid for small distances within an office, and accounts for some optimisation by the secretary. If the telex secretary is requested to walk to the mail box for fetching both a receipt and a new telex, then she only has to walk *once*.

$$D_{Secretary_bus} = walk \left[0.025 \right]$$

The total resource demand for walking is:

$$D_{Secretary_bus}^{Telex_secretary} = C_{Secretary_bus}^{Telex_secretary} \cdot D_{Secretary_bus}$$

$$D_{Secretary_bus}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 0.025 \cdot (1 + 10 + 2 \cdot 2) \\ 0.025 \cdot (2 \cdot 1 + 10 + 3 \cdot 2 + 1 \cdot 10) \end{bmatrix}$$

$$D_{Secretary_bus}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 0.375 \\ 0.70 \end{bmatrix}$$

10.3.1.5 Resource Demands for Secretary as Store

Since finding the appropriate addresses is more complex for external receivers than for internal receivers, this is reflected in the resource demands for `get_int_adr` and `get_ext_adr`.

The `store_log` operation includes time for (1) finding a log number and (2) writing down in a book for future reference (2 a) date received, (2 b) date sent, (2 c) topic, and (2 d) addressee of the telex. `get_log` is simpler.

$$D_{Secretary_store} = \begin{matrix} get_int_adr \\ get_ext_adr \\ store_arch \\ get_arch \\ store_log \\ get_log \end{matrix} \begin{bmatrix} 0.1 \\ 0.2 \\ 0.1 \\ 0.1 \\ 1.0 \\ 0.1 \end{bmatrix}$$

The total resource demand for the secretary as store is:

$$D_{Secretary_store}^{Telex_secretary} = C_{Secretary_store}^{Telex_secretary} \cdot D_{Secretary_store}$$

$$D_{Secretary_store}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 0.1 \cdot (2 + 1) + 0.1 \cdot 1 + 1 \cdot 1 \\ 0.1 \cdot 2 + 0.2 \cdot 9 + 0.1 \cdot 1 + 0.1 \cdot 1 + 1 \cdot 1 + 0.1 \cdot 1 \end{bmatrix}$$

$$D_{Secretary_store}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 1.4 \\ 3.3 \end{bmatrix}$$

10.3.2 Total Resource Demand for Telex Secretary

The total resource demand for the telex secretary is:

$$D^{Telex_secretary} = D_{Secretary_bus}^{Telex_secretary} + D_{Secretary_desk}^{Telex_secretary} + D_{Secretary_copy}^{Telex_secretary} + D_{Secretary_store}^{Telex_secretary}$$

$$D^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 3.6 \\ 30 \end{bmatrix}$$

This means that with the model and parameters selected, the telex secretary uses 3.6 minutes to handle an incoming telex, and 30 minutes to handle an outgoing telex.

10.3.2.1 Model Validation

The resulting resource demand for outgoing telexes corresponds to the 28 minutes as reported in the internal Statoil report [Sta94c]. 3.6 minutes for incoming telexes could be closer to the 5 minutes estimated for incoming telexes in the Statoil report [Sta94c]. Note that both the value 3.6 and the value 5 minutes were *estimated*. The first one (3.6 minutes) was estimated by me together with the contract coordinator JRS in Statoil, and the second value (5.0 minutes) had previously been estimated by the contract coordinator JRS.

A better correspondence would require further work with the parameters in the model. Such a validation process is supported both by the SP modelling language and the method with iterations as described in this thesis. Therefore, the method is applicable to performance modelling for routine secretary work.

With the workload $W_{Sales_Administration}^{Today}$ described in Section 10.1.4, the secretary will spend $20 \cdot 3.6 + 10 \cdot 30 = 372$ minutes ≈ 6.2 hours each day on telex handling. This corresponds roughly to a full-time position (7.5 hours each day), especially if we also consider the context changes which are necessary for each telex, and which are not considered explicitly. This high utilisation corresponds to the overloaded feeling of

the telex secretary. If we compare with a computer system, an average utilisation of 83 % ($\frac{6.2}{7.5} = 0.83$) is high. The high average utilisation is possible because humans are more flexible than computers. If the workload is too high, humans may perform shortcuts to get the work done. During shortfall she will use twice as much time, i.e. over 12 hours, which is too much for one person if she is to work with sufficient quality. Also under the anticipated, future load, it is impossible for a single telex secretary to cope with the workload.

The `type` operation for the outgoing telexes accounts for approximately half of the telex secretary workload. Therefore, a change in the parameters p_i , p_s , n_p , x or the resource demand for secretary `type` could change the secretary workload drastically.

10.3.2.2 Contention Modelling

Modelling of contention was not necessary in this case study, since the manual system was not going to be used when the computerised solution in Chapter 11 came in production. If contention modelling had been necessary, the static model gives us the parameters we need. As a simple illustration, an open two class queueing network with the parameters above ⁷ will predict that the secretary on the average had 0.9 ingoing and 3.8 outgoing telexes on here desk waiting to be handled. The average response time for incoming telexes would be 21 minutes. For outgoing telexes, the average response time would be 2 hours and 50 minutes. An open queueing network assumes independent arrival of telexes, which may be questionable. With more sophisticated contention models, other assumption could have been used.

10.3.2.3 Error in Input Data

The total resource demands described in Section 10.3.2 were initially higher. The initial estimate was:

$$D^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 4 \\ 43 \end{bmatrix}$$

The most important reason for this estimate was another value for the parameter x . The parameter x , the number of iterations between contract specialists and telex secretary, was estimated to be 2 by me. The distance from the telex secretary to the mail box, d_p , was initially 10, based on my estimations. After discussions with JRS we found $x = 1$ and $d_p = 2$ to be more appropriate and the results were as presented

⁷Following the notation from Section 3.2.1, $D_{in.telex} = \frac{3.6}{60} = 0.06$, $D_{out.telex} = \frac{30}{60} = 0.5$, $\lambda_{in.telex} = \frac{20}{7.5} = 2.7$, $\lambda_{out.telex} = \frac{10}{7.5} = 1.3$, $U_{in.telex} = \lambda_{in.telex} D_{in.telex} = 0.16$, $U_{out.telex} = \lambda_{out.telex} D_{out.telex} = 0.67$.

before. Identification of this error and subsequent corrections were simple because of the structured method which had been used.

10.4 Establish Components: Contract Specialists

The "Better Telex in Gas Sales Administration" project will affect the interaction between the contract specialist and the telex secretary. The `Contract_specialist` in Figure 10.4 is abstract and does not say anything about the distribution of work between the contract specialists in `Contract_coordination` and in `Troll`. `Contract_specialist` devolves work on `Contract_specialist_bus` as shown in Figure 10.8 below. The link from the `Contract_specialist` to `Contract_specialist_bus` is a communication link:

The contract specialist must walk to the telex secretary to take the telex, and the telex secretary walks with the typewritten copy of the telex back to the contract specialist for comments. Hence, there is a relation between the number of times the contract specialist walks and the number of times the telex secretary walks. Therefore, the `walk` is relevant for the contract specialist. As shown in the com-

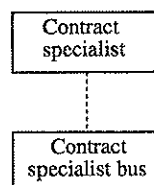


Figure 10.8: Rudimentary SP model for the present handling by the contract specialists.

plexity specification below, the walking operation is only used for `out_telexes`. For all the x iterations between the `Contract_specialist` and the `Telex_secretary`, the `Contract_specialist` walks the distance d_s , which is the distance between the `Contract_specialist` and the `Telex_secretary`:

$$C_{Contract_specialist_bus}^{Contract_specialist} = out_telex \begin{matrix} walk \\ \left[\begin{matrix} x d_s \end{matrix} \right] \end{matrix}$$

In this simple case, the complexity specification degenerates to only one complexity function. Replacing the parameters with the average numbers in Table 10.1 gives:

$$C_{Contract_specialist_bus}^{Contract_specialist} = out_telex \begin{matrix} walk \\ \left[\begin{matrix} 10 \end{matrix} \right] \end{matrix}$$

10.5 Evaluate and Validate Static Model of Contract Specialist

The work complexity functions for the contract specialists are used to estimate the performance of the contract specialists in this section. Section 10.5.1 estimates resource demands for the primary contract coordination operations, and in Section 10.5.2 the total resource demands for the contract specialist are estimated.

10.5.1 Resource Demands for Contract Specialists

Contract_specialist_bus only offers one primary operation, namely *walk*. As described in Section 10.1.3.1, the contract specialists also perform a lot of other work, which is not relevant in this context. The resource demands for the contract coordinator JRS for this operation, measured in minutes is taken from [Sta94c]:

$$D_{\text{Contract_specialist_bus}} = \text{walk} \left[0.025 \right]$$

The saving in typing for each contract specialist of 0.5 minutes for each telex in [Sta94c], is not taken care of there, because some additional work also must be performed by the contract specialists to use the “Better Telex in Gas Sales Administration” project.

10.5.2 Total Resource Demand for Contract Specialists

The total resource demand for the contract specialists is:

$$D^{\text{Contract_specialist}} = C_{\text{Contract_specialist_bus}}^{\text{Contract_specialist}} \cdot D_{\text{Contract_specialist_bus}}^T$$

$$D^{\text{Contract_specialist}} = \text{out_telex} \left[0.25 \right]$$

Using the workload vector $W_{\text{Sales_Administration}}^{\text{Today}}$ from Section 10.1.4, the contract specialist will spend $10 \cdot 0.25 = 2.5$ minutes on the telex which is affected by this case study. But since the total number of contract specialists in Contract coordination and Troll is 12, the total time during a typical day is $12 \cdot 2.5 = 30$ minutes. This is only a fraction of the time which is spent by the telex secretary

10.6 Summary of Findings

The overall conclusion is that the static model came out with estimates which were consistent with observed behaviour, in terms of utilisation. The correspondence between the model and what was expected was good enough to be used as a basis for *changes* in the actual system. The parameters of the model could have been refined if the model should have been used for example for response time estimations, which are more sensitive to parameter quality. The error in estimating the distance d_p in Section 10.3.2.3 showed the model to be quite sensitive to office layout, i.e. the distances between copy machines and desks and case workers.

The presentation in this chapter gives a top-down impression of what happened during the modelling of the organisation in the gas sales telex administration. According to such a top-down approach, for example the system boundaries are determined before the system is modelled. In practice, modelling is more complex. Details will often affect general matters in a bottom-up manner. The method of this thesis is iterative to take account of this.

The overall scientific objective of this case study was described in Section 10.1.1 as to sort of the essential problems from the more academic ones. The issues may be structured in the following way. Firstly, is it *at all possible* to model human workflows, with no constraints on the size of the model, i.e. without any aggregations of operations or structural variables.⁸ The cost of making and maintaining such a model would in most cases be too large in practice (c.f. the tradeoff between accuracy and cost in Section 5.4.5.2). Therefore a *practical* model must aggregate the operations and structural variables of the real world. This leads to the second question. Is the inaccuracy introduced by the chosen aggregations acceptable?

Given that it is possible to create a reasonably correct model with manageable size, the third question is if the accuracy in the computation of the model is reasonable. Is the accuracy of the model solution method acceptable? In summary the three questions may be formulated as:

F_1^G : Is it possible to model human behaviour at all?

F_2^G : Is the inaccuracy introduced by the chosen aggregations acceptable?

F_3^G : Is the accuracy of the model solution method acceptable?

Question F_1^G , about the possibility of making accurate estimates for human workflows, would be positively answered for fairly routine workflows. If the workflow is routine or stable, then it is possible to find a mean. The problems in identifying the

⁸Table 10.1 shows structural variables for the manual solution of the Gas Sales Telex Administration Case Study.

mean is similar to the problems in estimating the mean for computer systems. Statistical techniques are needed. As an example of the similarity between manual and computer operations, see the discussion about the operation walk in Section 10.3.1.3.

The routine subprocesses in a template workflow will also be possible to estimate. For ad-hoc workflows, it is (per definition) not possible to say anything general about resource demands. Attempts to model the skilled part of the contract specialist work showed that for this almost ad-hoc type of work, tradeoffs with other factors must be included in the method (c.f. Section 8.1.1). This is outside the scope of this thesis. These other factors are much more fixed in routine work of the telex secretary, and may therefore be ignored during modelling of the routine work for the telex secretary.

For the second question, F_2^G , it is illuminating to compare alternative aggregations. The model in Chapter 10 is more detailed than the coarse model in the report [Sta94c]. A detailed model has the potential to be more accurate, if the input data is more accurate. If parameter accuracy is not sufficient, it is easy to aggregate the model before using it, producing a coarser model which is more suited to the available parameters. At the one extreme the two operations `in_telex` and `out_telex` could have been measured directly, in terms of the time used. This coarse approach is used for the human workflows in Chapter 11. There are also other approaches in between the coarse and the more detailed approach selected in this chapter. Selecting a suitable granularity for modelling of human workflows is no different from selecting granularity when modelling a computer system.

The level of detail represented by the model was influenced by the use of SP in software modelling. It was chosen partly to explore the correspondence between human and computerised information processing. The two are compared in Section 11.7 which illustrates the generality of the SP paradigm. This level of detail enables reuse of the static model components, when parameters representing e.g. office layout, skill levels, and information processing requirements vary.

Modelling the walking operations gave challenges. Estimating distances walked as in Table 10.1 was decided upon as described in Section 10.3.1.4. A problematic area is how the secretary will combine walking with several items to the same place: a smart secretary might carry several items in one go. Another issue is whether she could think while walking. As a possible solution, the legs and brain could be modelled as separate resources⁹ Some tasks occupy both resources, like sitting by the desk and reading, while other tasks like walking with some documents will only occupy the legs, leaving the brain free for other purposes.

A more detailed model is more suited for reuse. The SP model shown in Figure 10.7 may be the beginning of a more general model of case workers, because communication, processing and storage are the basic ways of handling information. In a more general SP model, the processing operations will be more complex. The major difference between the telex secretary and most case workers is that case workers have a larger number of different processing operations compared to the telex secretary,

⁹This level of detail also has its humorous sides...

who only performs relatively few different processing operations.

For the third question F_3^G , for this case study, there are no built in assumptions in the SP calculations which would give rise to inaccuracy. For contention modelling, the usual considerations for approximate solution methods would apply. In general, development of a performance model is a tradeoff between accuracy and cost (c.f. Section 5.4.5.2). For general software it may be possible to divide the development cost for a performance model between several installations. Sometimes the accuracy of a general model may be sufficient. In cases where the accuracy of a general model is too poor, it may be possible to make a customised model. The customisation may hamper the reusability of the model, and will therefore increase the cost of the model. The cost of performance modelling may therefore on some occasions be too large in practice.

Chapter 11

Application to a Workflow Computer System

This chapter presents an outline of an overall performance model of Lotus Notes, which forms the basis for the computerised workflows in the gas sales telex administration sector in Statoil. A work model of Lotus Notes will be of interest for performance engineering. Lotus Notes work models could also be useful for capacity planning and tuning in large organisations. This chapter is based on talks with personnel in Statoil and the latest documentation [Sta94b, Sta94c]. The model of the client-server Lotus Notes architecture is based on work done by Alexander Kowalski [Kow95], who was a diploma student under my supervision.

To limit the amount of work, replication in Lotus Notes is not modelled in this thesis. If a model of replication in Lotus Notes had been available, it could have been included in the overall Lotus Notes model.

Note that the existing platform in this chapter is not existing in the same sense as the existing organisation in Chapter 10. The manual information system is available in the gas sales telex administration sector, while Lotus Notes is existing somewhere else in Statoil (or in another company), but not in the gas sales telex administration sector.

The background for this case study was explained in Chapter 10. This chapter is structured as follows: Section 11.1 describes the requirements for the Lotus Notes system. Section 11.2 explains a work model of basic Lotus Notes functionality, while Section 11.3 gives an overall work model of Lotus Notes replication. Section 11.4 describes the integration of the workflow platform Lotus Notes with the manual system in Chapter 10. The interaction between the telex secretary and Lotus Notes is described in Section 11.5. The complete work model for the computerised solution is evaluated in Section 11.6. Section 11.7 compares the SP model of the computerised solution with the SP model for the manual solution. A generalisation of the Gas Sales Telex Administration Case Study is discussed in Section 11.8.

11.1 Specify System Requirements

11.1.1 Objectives of Performance Model

The objective in Section 10.1.2 was:¹

O_1^G : Find out if response time was smaller for the computerised solution than for the manual solution.

This objective will be detailed in this chapter. Assessment of the organisational benefit of using Lotus Notes is important. When a large organisation installs workflow systems, they cannot afford to buy unsuitable equipment, because it is extremely costly and time-consuming to do fix-it-later with thousands of users involved. Lotus Notes for the whole gas division will typically cost about 4 MNOK, 2500 NOK per workstation per year. The total Statoil organisation has 5000 Notes clients. In this context, a performance objective is:

$O_{1,1}^G$: Find the organisational benefit of using Lotus Notes.

This objective may be seen as an extension of objective O_1^G . In addition to the performance objectives in Section 10.1.2, capacity planning for the Lotus Notes application is also relevant. Examples of relevant questions are: (1) For what applications should Lotus Notes be used? (2) Which hardware is necessary for Lotus Notes? (3) What will be the performance consequences of replacing the Lotus Notes database with a full-blown Oracle relational database? (4) How do we detect the problem when applications are designed in a local setting and later used in a network setting, with poor performance as the result? Thus, a second performance objective is:

O_2^G : Find the consequences of replacing parts of the computerised information system with other components.

11.1.2 Determine System Boundaries

Replication needs to be introduced before two important subsystems are described in Section 11.1.2.2.

¹Superscript G in O_1^G refers to the Gas Sales Telex Administration Case Study to avoid confusion with the Blood Bank Case Study.

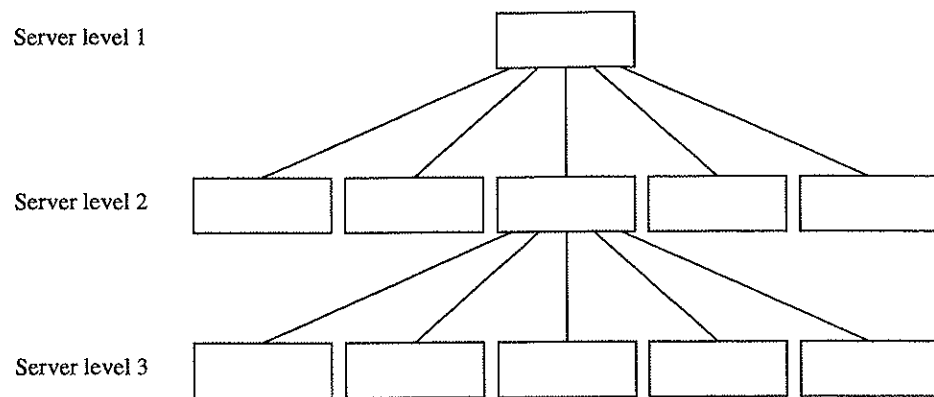


Figure 11.1: Three server levels in Statoil

11.1.2.1 Replication in Lotus Notes

Lotus Notes distributes the database across several servers. This improves performance since the user can use the server which is closest to him. Each server holds a copy of the shared database. Databases on different servers are kept consistent through replication. During replication, changes are propagated between all the servers. As the result of replication, all databases are equal (at least in the simple case — partial replication is also possible).

Statoil has designed a special replication method for Lotus Notes. This method builds on the server hierarchy in Figure 11.1, ² and is explained in Appendix A. As depicted in Figure 11.1, each server at server level 1 has servers at server level 2 as nodes, and servers at server level 2 have nodes at server level 3.

For every person in Statoil, basic information like name and office number is stored in the name and address book which is stored at server level 1. The servers at server level 1 are maintained in Stavanger [Sta93]. Each department or office should always have at least one server at server level 2. For small offices, where no server level 3 servers are needed, the server level 2 server must also work as a mail and database server. Normally, server level 3 works as mail and database servers.

11.1.2.2 Two subsystems

Figure 11.2 shows how two interacting subsystems can be identified in the Lotus Notes system. The Replication subsystem takes care of the replication of documents between the hierarchy of Lotus Notes servers in Figure 11.1, while the Client-server subsystem deals with the interaction between a `Server_level_3_server` and the

²This figure is not an SP model. An SP model represents logical structure of work, whereas this model represents physical server structure.

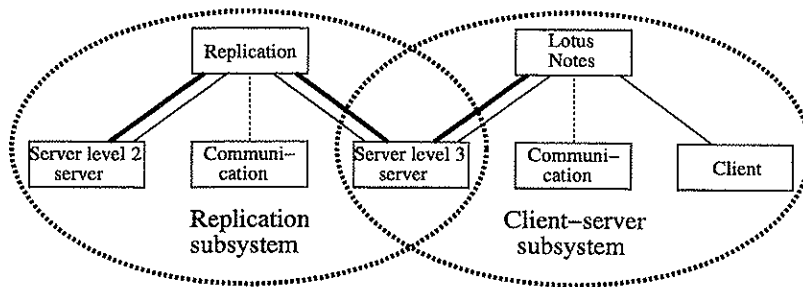


Figure 11.2: Separate parts of Lotus Notes modelling

Client. For simplicity, only the replication between server level 3 and server level 2 is considered in this section. The replication between server level 2 and 1 is approximately similar to the replication between server level 3 and 2. Both the `Server_level_2_server` and the `Server_level_3_server` are of course shared with other users and other applications. This figure will be expanded in later sections. Both `Server` and `Client` modules will have several primary resources: i.e. disk and CPU. Finally, the `Communication` module may be more complex.

In this case study, the server and the network are shared resources, whereas the client is not shared. Thus, only the server and the network are potential performance bottlenecks which are subject to analysis in this thesis. To limit the complexity, the server (and not the network) is the focal point of the case study.

11.1.3 Estimate Workload

In Section 8.2.4, a `document` and a `view` were identified as the major data structures in Lotus Notes. Both a `document` and a `view` can be *opened*. It is only meaningful to *store* a `document`, but not meaningful to store a `view`. Therefore, three typical Lotus Notes operations were modelled:

```

open_view  Open a view showing the content of a database
open_doc   Open a document which is in a database
store_doc  Store a document in a database

```

These three operations correspond to the routine work of a case worker. Of course, other operations for example searching for a document, or database maintenance will also be used, but in this case study, such operations will be relatively rare. For these three operations, two parameters were investigated:

- s* Size of the documents stored in the database measured in bytes (a byte will roughly equal one character).
- n* Number of documents stored in the database.

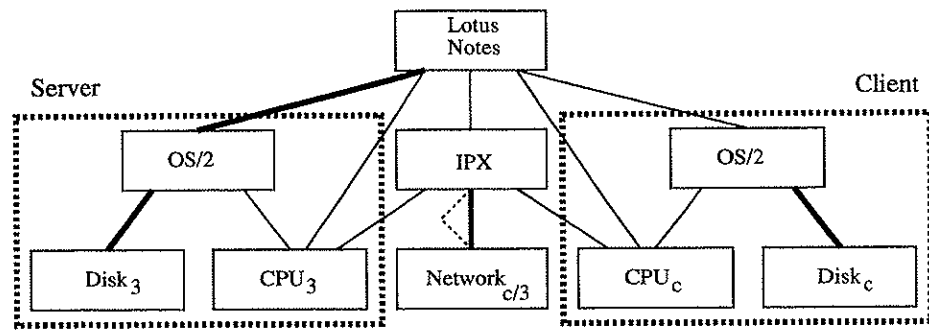


Figure 11.3: SP model for Lotus Notes

11.1.4 Determine Performance Requirement

The performance requirements for the “Better Telex in Gas Sales Administration” project were specified in Section 10.1.5. These performance requirements will also be used here. In addition, performance requirements for the Lotus Notes application corresponding to O_2^G in Section 11.1.1 may also be developed.

11.2 Establish Components: Lotus Notes Client-server

Lotus Notes devolves work both on the server level 3 server (termed Disk_3 and CPU_3) and on the client (termed Disk_c and CPU_c) as shown in the SP model in Figure 11.3.

With the available measurement tool SPM/2 from IBM, it was not possible to measure the component OS/2 in Figure 11.3 [Kow95]. SPM/2 was designed for another network than used in our installation. Therefore, measurements for the network component were also hard to get.³ The resulting, reduced model is shown in Figure 11.4. Using hardware monitors and better measurement tools,⁴ the measured SP model could resemble the model in Figure 11.3.

For Kowalski, it took roughly three months to do the measurements. With more practice, this time could have been significantly reduced. On the other hand, Kowalski also noted that more measurements should be done to increase the confidence in his results. In the ideal case, the vendor should include a complexity specification

³With hardware monitors it was easy to measure the network component in Statoil. Kowalski did not have access to such tools.

⁴IBM also purchases more expensive and more sophisticated measurement tools. Sophisticated measurement tools may also be available from other vendors.

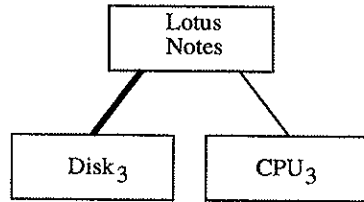


Figure 11.4: The SP model measured by Kowalski

for his software. This is possible since work characterisation is workload independent. For the vendor of off-the-shelf products like Lotus Notes, it will be feasible to make detailed work specifications. See Section 5.4.5.2, which is based on Vetland's thesis [Vet93], for more discussion.

The resulting complexity specification for the relation between the three user operations above and instructions on the Intel Pentium 90 MHz server CPU with 256 KB cache is:

$$C_{CPU_3}^{Lotus_Notes} = \begin{matrix} & & instruction \\ open_view & & \\ open_doc & & \\ store_doc & & \end{matrix} \begin{bmatrix} 1\,580\,s + 3\,000\,n + 62\,000\,000 \\ 538\,s + 15\,000\,n + 18\,000\,000 \\ 95\,s + 26\,000\,000 \end{bmatrix}$$

The parameter s is the average size of documents stored in the database measured in bytes and the parameter n is the number of documents stored in the database. Thus, an `open_view` on a database with 2 000 documents and with a size of 5 kB for each document is equivalent to:

$$\begin{aligned} \text{CPU instructions} &= 1\,580 \cdot 5 \cdot 1\,000 + 3\,000 \cdot 2\,000 + 62\,000\,000 \\ &= 7\,900\,000 + 6\,000\,000 + 62\,000\,000 \\ &= 76\,000\,000 \text{ instructions} \end{aligned}$$

The complexity specification for the relationship between the three user operations and the read sector and write sector operations on the 1 GB Quantum Empire SCSI disk on the server (with 32 MB primary memory on the server and with 16 MB of primary memory on the client) is expressed as follows:

$$C_{Disk_3}^{Lotus_Notes} = \begin{matrix} & & read_sector & & write_sector \\ open_view & & & & \\ open_doc & & & & \\ store_doc & & & & \end{matrix} \begin{bmatrix} 0.0067\,s + 0.167\,n + 86 & 112 \\ 0.0012\,s + 8.325 & 42 \\ 9 & 0.00404\,s + 132 \end{bmatrix}$$

An open_view of 2 000 documents with a size of 5 kB each will devolve:

$$\begin{aligned}\text{Physical sectors read from disk} &= 0.0067 \cdot 5 \cdot 1\,000 + 0.167 \cdot 2\,000 + 86 \\ &= 34 + 334 + 86 \\ &\approx 450 \text{ physical sectors}\end{aligned}$$

11.2.1 Validation

In regression analysis the coefficient of determination R^2 can theoretically vary between 0 and 1 [Jai91]. The closer to 1, the better the regression. In Kowalski's measurements, R^2 ranged between 0.84 and 0.9995. For some of the measured parameters, confidence intervals were broad. Therefore these parameters could not be used outside of some boundaries, e.g. the number of documents have to be between 2000 and 5000. The regression could have been better if more measurements had been done, or if the work model had been refined. Making a detailed work model is hard because of the unknown behaviour of the software. Contact with the vendor, Lotus, will be necessary to make a really good work model, but even then the inherent problems of work modelling as mentioned in Section 5.4.5.2 still apply.

This work model depends quite strongly on the selected disk. A refined model, where the operating system OS/2 is modelled as an extra software layer as described in Figure 11.3, would make the work model more hardware independent. The measurements may also be sensitive to size of CPU cache, client primary memory, server cache, server primary memory, and to the version and type of operating system and the version of Lotus Notes. These installation parameters could be included as a parameter in the complexity specifications, making them more general.

More careful modelling would increase the reproducibility of the measurements, e.g. changes in fragmentation and placement of files on the disk could be taken care of by including all relevant states and then estimating a suitable average. Changes in fragmentation or file placement may also be taken care of by changing the workload specification by for example formulating new operations for some particularly important states. This would complicate the modelling task, but give more accurate results. As another example of state dependency which may need extensions in the workload specification, it is known that opening of a database view is more complex in Lotus Notes if there has been a change in database indexes, than if indexes have not been changed. The robustness of the complexity specification to changes like internal Lotus Notes tuning parameters could also be studied more carefully.

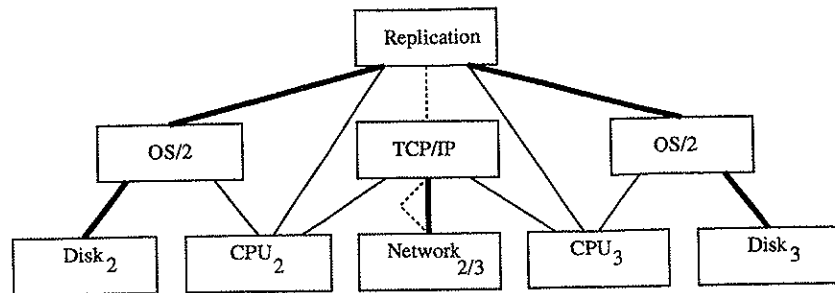


Figure 11.5: Replication architecture for Lotus Notes.

11.3 Establish Components: Replication Architecture

Replication in Statoil was introduced in Section 11.1.2.1, and elaborated upon in Appendix A. Figure 11.5 shows an outline of an SP model for replication. In this figure, the resources which are used during replication between server level 2 and server level 3 are shown.

Note the distinction between the SP hierarchy levels and the replication hierarchy levels in Figure 11.5. $Disk_2$ and $Disk_3$ are at the same SP hierarchical level, but at a different replication hierarchical level. The archives will be located in $Disk_3$. Several of the modules will be more complex than what is shown here. For example, inside of the $Network_{2/3}$ components there are also gateways. And if there are large differences between the servers at server level₃, this must be indicated with separate models.

Each module will also have operations. The replication module in Figure 11.5 may have only one operation `perform_replication`. Depending on the size of the archive (which is specified in an extent specification as described in Section 4.6.1), this operation will result in a number of `read_sectors` on $Disk_3$ and `send_packets` on $Network_{2/3}$. Since replication is a never-ending-loop with three hour intervals, only two scenarios are possible on the server. Either replication happens or replication does not happen. If replication has priority over the client-server work in Section 11.2, load concealment [LZGS84] can be assumed with some inflation factor $I_{repl} > 1$, so the refined service demand D' is increased compared to the original resource demand D :

$$D' = I_{repl}D$$

The replication architecture between server level 1 and server level 2 will be quite similar to the replication architecture in Figure 11.5 between server level 2 and server level 3.

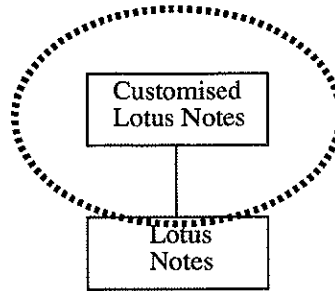


Figure 11.6: Simplified, customised Lotus Notes model.

11.4 Establish Components: Customising Lotus Notes

Lotus Notes is customised in the “Better Telex in Gas Sales Administration” project. As illustrated in Figure 11.6, this customisation is put on top of the generic Lotus Notes work model described in Figure 11.3 and is indicated with the dotted ellipse in the figure. In practice, more complexity specifications than $C_{\text{Customised.Lotus.Notes}}$ are needed.⁵

As described in Section 8.2.4, the two basic data structures in Lotus Notes are *view* and *form*. In the gas sales telex administration information system, several views and forms exist. These twelve operations are offered to the users:

view_distribution_list: Show a view of all the distribution lists. To simplify the process of sending a telex to several receivers, *distribution lists* are made and maintained. Each distribution list has a name and consists of several addressees. An addressee has a name, a telex number and possibly a Notes address.

distribution_list_form: Open one particular distribution list.

view_user_profile: Show a view of all the user profiles. Each sender of telexes has a *user profile* which contains his name, office number, his reference and all the companies he sends the telex on behalf of. If for example Statoil is an operator of an area, then telexes may be sent on behalf of the other partners, e.g. Total, Elf or Saga.

user_profile_form: Open one particular user profile.

⁵For example, the last two operations `send_telex_internally` or `send_telex_externally` in $C_{\text{Customised.Lotus.Notes}}$ will use the `IPX.c/3` component. This may be specified by the $C_{\text{Customised.Lotus.Notes}}$ complexity specification. When estimating these two operations, care should be taken to also include the telex receipt.

view_mail_list: Show a view of all mail lists. For internal receivers, there is also a *mail list*.

mail_list_form: Open one particular mail list.

view_telex: Show a view of all telexes which are either incoming, sent, outgoing or under work (Initially, there were three views for viewing telexes, discriminating between incoming, sent and outgoing telexes. To be more user friendly, these three views were concatenated into one view, which is termed **view_telex**.)

telex_form: Open one particular telex for editing.

store_telex_form: Store a telex which is sent.

show_stamp: Show the stamp on a telex. An electronic stamp is used for each telex. This stamp contains the archive number, subject, status, sent date, archive date, receiver, external copy to, internal copy to and receipts received. Most often this stamp is invisible, with the exception that it is visible for an archived telex and for a sent telex which is not archived.

send_telex_internally: Send a telex within the sales administration sector in Statoil.

send_telex_externally: Send a telex outside of the sales administration sector in Statoil.

Operations for changing and storing distribution list, user profiles and mail lists are also needed, but if these lists are seldom changed, the corresponding operations may be ignored in a coarse modelling. Below, the complexity specification for the link from *Customised_Lotus_Notes* to *Lotus_Notes* is outlined:

		<i>open_view</i>	<i>open_doc</i>	<i>store_doc</i>
<i>Customised_Lotus_Notes</i> = <i>Lotus_Notes</i>	<i>view_distribution_list</i>	1		
	<i>distribution_list_form</i>		1	
	<i>view_user_profile</i>	1		
	<i>user_profile_form</i>		1	
	<i>view_mail_list</i>	1		
	<i>mail_list_form</i>		1	
	<i>view_telex</i>	1		
	<i>telex_form</i>		1	
	<i>store_telex_form</i>			1
	<i>show_stamp</i>		1	
	<i>send_telex_internally</i>		1	1
	<i>send_telex_externally</i>		1	1

As seen from this complexity specification, only a fraction of the elements in this 12×3 matrix needs to be estimated. These approximate estimates should be replaced by measurements as soon as possible.

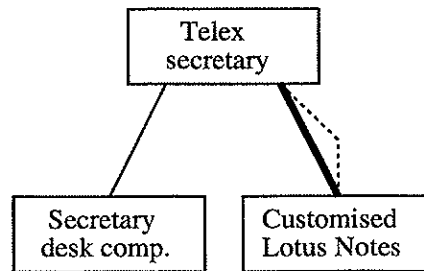


Figure 11.7: SP model which shows how the secretary and Lotus Notes are used.

11.5 Establish Components: Secretary Using Lotus Notes

Figure 11.7 shows how Lotus Notes is used by the `Telex_secretary` in the Lotus Notes solution. The `Telex_secretary` also uses operations from `Secretary_desk_comp`, in the same way as in the manual solution in Chapter 10. `Telex_secretary` uses `Customised Lotus Notes` as memory and communication and `Secretary_desk_comp` as processing. This is shown by the three links in Figure 11.7. `Secretary_desk_comp` is used for processing because the secretary for example will type on the client. Most of the `Customised Lotus Notes` operations are memory operations. The `send_telex_internally` and `send_telex_externally` operations of `Customised Lotus Notes` are communication operations.

With the Lotus Notes solution, the secretary only has to `type_characters`. Of course, she will also need to think and will also use herself as a storage system, but this activity has not explicitly been modelled in the manual solution, and will therefore not be explicitly modelled here either. The resulting complexity specification is shown below:⁶

$$C_{Secretary_desk_comp}^{Telex_secretary} = \begin{matrix} & & type_char \\ in_telex & \left[\begin{array}{c} 10 \\ 100 \end{array} \right] \\ out_telex & \end{matrix}$$

When Lotus Notes is introduced in the organisation, paper-based workflow will still be used during a transition period. However, this section only looks on the situation when the secretary performs all the telex handling with the Lotus Notes solution. If she still uses paper-based workflow, the resulting model will be a superset of this model and the model for the paper-based solution in Section 10.2.

⁶This complexity specification could of course be refined. It would for example be possible to break down the number of characters needed to perform each Lotus Notes command, e.g. `view_distribution_list` and `distribution_list_form`.

As depicted in Figure 11.7, the `Telex_secretary` will also use `Lotus_Notes`. Because of all the `Lotus_Notes`-operations, this complexity specification is divided in three. The first part is shown below:

$$C_{\text{Telex_secretary}}^{\text{Customised_Lotus_Notes.1}} = \begin{matrix} \text{in_telex} \\ \text{out_telex} \end{matrix} \begin{bmatrix} \text{view_distr_list} & \text{distr_list_form} & \text{view_user_profile} & \text{user_profile_form} \\ 1 & 1 & 0 & 0 \\ 1+y & 1+y & 0 & 0 \end{bmatrix}$$

Note that the user profile is not used during normal operation, but only when there is a need to create new distribution lists. In Section 11.4, this was described as not being of importance here. The parameter y in the complexity specification is explained in Table 11.1. y measures the number of drafts which are transported between the telex secretary and the contract specialist in the computerised solution. If y increases with the computerised solution compared to x in the manual solution, this may mean that the (local) quality of the telexes is improved. With respect to the (global) quality of work in the gas sales administration, this local improvement in quality may *not* lead to global quality improvements (c.f. discussion of efficiency and effectiveness in Section 2.4).

The mailing lists are used both for incoming and for outgoing telexes. For outgoing telexes, the iterations y also affect this second part of the complexity specification:

$$C_{\text{Telex_secretary}}^{\text{Customised_Lotus_Notes.2}} = \begin{matrix} \text{in_telex} \\ \text{out_telex} \end{matrix} \begin{bmatrix} \text{view_mail_list} & \text{mail_list_form} & \text{view_telex} & \text{telex_form} \\ 1 & 1 & 1 & 1 \\ 1+y & 1+y & 1+y & 1+y \end{bmatrix}$$

The stamp of the telex is referenced both for incoming and for outgoing telexes as shown in the third and final part of the complexity specification below:

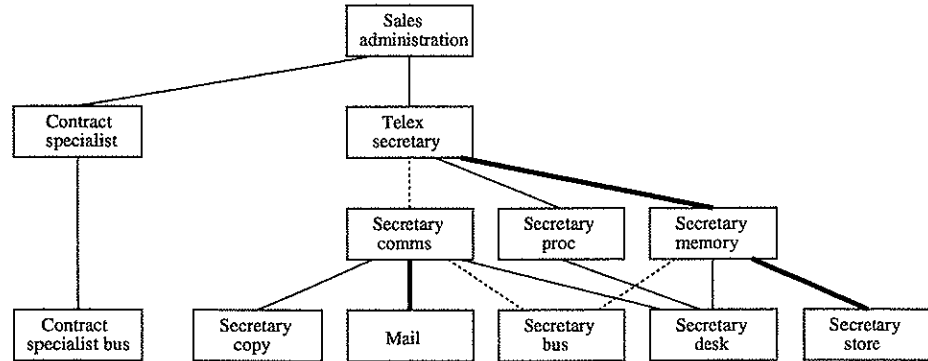
$$C_{\text{Telex_secretary}}^{\text{Customised_Lotus_Notes.3}} = \begin{matrix} \text{in_telex} \\ \text{out_telex} \end{matrix} \begin{bmatrix} \text{store_telex_form} & \text{show_stamp} & \text{send_telex_internally} & \text{send_telex_externally} \\ 1 & 1 & 1 & 0 \\ 1+y & 1+y & y & 1 \end{bmatrix}$$

In this complexity specification, it is also indicated that the telex needs to be stored. Whereas an incoming telex only needs to `send.telex.internally`, an outgoing telex only needs `send.telex.externally`.

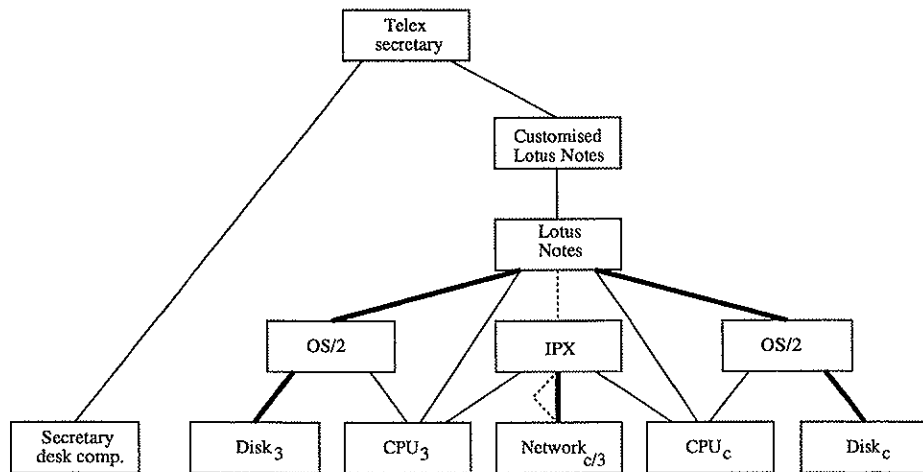
11.6 Evaluate and Validate Static model for Computerised Solution

Figure 11.8 shows an almost complete resource model for the manual and computerised solution. Note that the contract specialist is missing in the computerised solution. This is because the work which was performed by the component

11.6. Evaluate and Validate Static model for Computerised Solution181



a) Manual telex solution

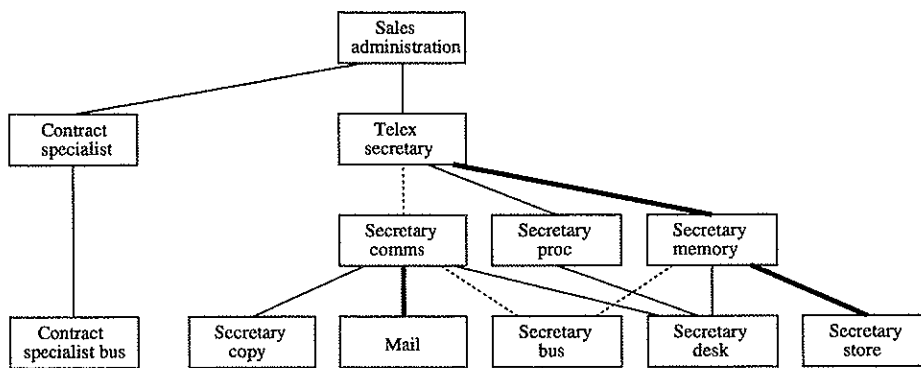


b) Computerised telex solution

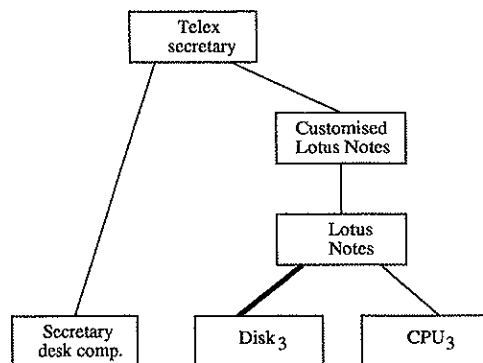
Figure 11.8: A “complete” SP model for (a) the manual and (b) the computerised solution.

Contract_specialist in the manual solution, is done by the network in the computerised solution. The primary Secretary resource in Figure 10.7 is not shown in Figure 11.8, because this resource was only necessary because the secretary actually performed all the work of the primary resources in Figure 10.7.

A computerised version of $D^{Telex.secretary}$ in Section 10.3.2, would answer the questions in Section 10.1.5. To derive $D^{Telex.secretary}$ for the computerised solution we would need complexity matrixes for all the components in Figure 11.8. Time was not available to parameterise the whole computerised platform. The server components $Disk_3$ and CPU_3 were measured in Section 11.2. In addition, the Secretary_Desk component was *estimated* in Section 11.5. The client components $Disk_c$ and CPU_c and the network component $Network_{2/3}$ and the operating system component $OS/2$ were not measured. The model which were parameterised in this thesis is shown in Figure 11.9.



a) Manual telex solution



b) Computerised telex solution

Figure 11.9: The model which was parameterised for (a) the manual and (b) the computerised solution.

11.6. Evaluate and Validate Static model for Computerised Solution 183

11.6.1 Resource Demands for Computerised Solution

The resource demands for the computerised solution which was parameterised is estimated in this section. It is necessary to replace the parameter name y by a number. Table 11.1 below shows the average number which is used. In the rest of this chapter, this number is used to calculate numeric estimates for resource demands.

Parameter	Value	Description of parameter
y	0.5	Number of iterations of telex drafts between the contract specialist and the secretary with the Lotus Notes solution. (The symbol x was used for the same purpose in the manual solution in Table 10.1 in Section 10.3.1.)

Table 11.1: Parameter representing average numbers for the Lotus Notes solution.

11.6.1.1 Resource Demands for Secretary as Desk

As a processor, the telex secretary types characters. The speed for the `type_char` operations is estimated to 0.02 minutes per character. Compared to the `type` operation in Section 10.3.1.3, which was for one page, this means that each page has 500 characters on the average. In the Gas Sales Telex Administration Case Study the number of characters in each page is estimated to be 1000 (c.f. Table 10.1 and Section 4.6.1). `type_char` is a bit slower to take account of the think time for the secretary when she switches between different windows.

$$D_{Secretary_desk_comp} = type_char [0.02]$$

All the comments on multiprogramming and nonlinearity in Section 10.3.1.3 will also apply to the `type_char` operation. Total resource demands for the secretary as desk in the computerised solution will be:

$$D_{Secretary_desk_comp}^{Telex_secretary} = C_{Secretary_desk_comp}^{Telex_secretary} \cdot D_{Secretary_desk_comp}$$

$$D_{Secretary_desk_comp}^{Telex_secretary} = \frac{in_telex}{out_telex} \begin{bmatrix} 10 \cdot 0.02 \\ 100 \cdot 0.02 \end{bmatrix}$$

$$D_{\text{Secretary_desk_comp}}^{\text{Telex_secretary}} = \begin{matrix} \text{in_telex} \\ \text{out_telex} \end{matrix} \begin{bmatrix} 0.2 \\ 2.0 \end{bmatrix}$$

11.6.1.2 Disk Server Resource Demands

Reading one sector on the disk is estimated to take 1 millisecond, and writing one sector is estimated to take 2 milliseconds.⁷ The service demand in minutes will therefore be:

$$D_{\text{Disk}_3} = \begin{matrix} \text{read_sector} \\ \text{write_sector} \end{matrix} \begin{bmatrix} \frac{1.6}{60000} \\ \frac{1.7}{60000} \end{bmatrix}$$

This is a rough estimation which is sensitive to physical placements of files on the disk and to the size of the files which are read and written. For more accurate and complex models of disks, see for example [RW94]. The resulting resource demands on the disk server is:

$$D_{\text{Disk}_3}^{\text{Telex_secretary}} = C_{\text{Customised_Lotus_Notes}}^{\text{Telex_secretary}} \cdot C_{\text{Lotus_Notes}}^{\text{Customised_Lotus_Notes}} \cdot C_{\text{Disk}_3}^{\text{Lotus_Notes}} \cdot D_{\text{Disk}_3}$$

An intermediate result is the matrix $C_{\text{Lotus_Notes}}^{\text{Telex_secretary}}$ below:

$$C_{\text{Lotus_Notes}}^{\text{Telex_Secretary}} = \begin{matrix} \text{in_telex} \\ \text{out_telex} \end{matrix} \begin{matrix} \text{open_view} & \text{open_doc} & \text{store_doc} \\ \begin{bmatrix} 3 & 5 & 2 \\ 4.5 & 7.5 & 3 \end{bmatrix} \end{matrix}$$

Using this intermediate result, and setting $n = 2000$ and $s = 5$ kB gives:

⁷These estimates are based on the equation:

$$\text{Service_demand} = \frac{\text{Seek_time} + \text{Latency}}{\#_sectors} + \text{Transfer_time}$$

The size of a typical telex was 5 kB. With a sector size of 0.5 KB, $\#_sectors \approx 10$ (remember 1 KB = 1024 bytes and 1 kB = 1000 bytes). Based on [Qua96], the seek time is estimated to be 9.5 ms for read and 11.0 ms for write, the average latency 5.6 ms and the transfer time 0.05 ms. This gives for read $\text{Service_demand} = \frac{9.5+5.6}{10} + 0.05 \approx 1.6$ ms.

11.6. Evaluate and Validate Static model for Computerised Solution185

$$D_{Disk_3}^{Telex_Secretary} = \frac{in_telex}{out_telex} \begin{bmatrix} 1500 \cdot \frac{1.6}{60000} + 850 \cdot \frac{1.7}{60000} \\ 2200 \cdot \frac{1.6}{60000} + 1300 \cdot \frac{1.7}{60000} \end{bmatrix}$$

$$D_{Disk_3}^{Telex_Secretary} = \frac{in_telex}{out_telex} \begin{bmatrix} 0.06 \\ 0.09 \end{bmatrix}$$

11.6.1.3 CPU Resource Demands

The processor is estimated to execute 70 million instructions per second.⁸ The resulting service demand per minute is:

$$D_{CPU_3} = instructions \left[\frac{1}{60 \cdot 70\,000\,000} \right]$$

The resulting resource demands on the CPU on the server is:

$$D_{CPU_3}^{Telex_secretary} = C_{Customised.Lotus.Notes}^{Telex.secretary} \cdot C_{Lotus.Notes}^{Customised.Lotus.Notes} \cdot C_{CPU_3}^{Lotus.Notes} \cdot D_{CPU_3}$$

Using the intermediate result in Section 11.6.1.2 and setting $n = 2000$ and $s = 5$ kB gives:

$$D_{CPU_3}^{Telex_Secretary} = \frac{in_telex}{out_telex} \begin{bmatrix} 540 \cdot \frac{1}{70 \cdot 60} \\ 800 \cdot \frac{1}{70 \cdot 60} \end{bmatrix}$$

$$D_{CPU_3}^{Telex_Secretary} = \frac{in_telex}{out_telex} \begin{bmatrix} 0.13 \\ 0.19 \end{bmatrix}$$

⁸Kowalski used the number 70 MIPS from the Dhrystone benchmark for the Pentium 90 MHz CPU [Wei96] when converting the resource demand from the SPM/2 measurement tool to number of instructions. Service demands for the CPU should be used with care if the actual CPU do not scale with respect to the Dhrystone benchmark, for the workload in this case study. The Dhrystone benchmark do for example not exercise I/O performance [Jai91].

11.6.2 Total Resource Demands for Computerised Solution

The total resource demand for the computerised solution is:

$$D^{Telex_secretary} = D_{Secretary_desk_comp}^{Telex_secretary} + D_{Disk.3}^{Telex_secretary} + D_{CPU.3}^{Telex_secretary}$$

$$D^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 0.4 \\ 2.3 \end{bmatrix}$$

This means that with the model and parameters selected, it takes 0.4 minutes to handle an incoming telex, and 2.3 minutes to handle an outgoing telex. Most of this time will be spent by the secretary using the user interface.

11.6.3 Performance Requirements Satisfied?

In Section 10.1.5, two performance requirements were formulated for the Gas Sales Telex Administration Case Study. The first performance requirement under present workload was:

$$P_1^G: R_{Computerised, Today, Normal}^{Sales_Telex_Administration} < R_{Manual, Today, Normal}^{Sales_Telex_Administration}$$

The resource demand for the computerised solution was estimated above as in Section 11.6.2:

$$D_{Computerised}^{Sales_Telex_Administration, Today, Normal} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 0.4 \\ 2.3 \end{bmatrix}$$

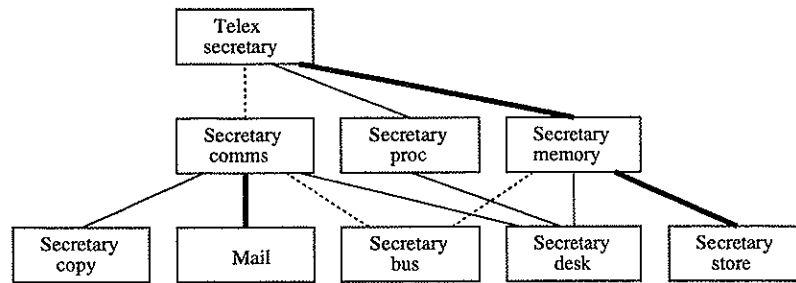
The resource demand for the telex secretary and the contract specialist in the manual solution were estimated in Section 10.3.2 and 10.5.2 respectively. We have to add them together to get the total resource demand for the manual telex administration:

$$D_{Manual, Today, Normal}^{Sales_Telex_Administration} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{bmatrix} 3.6 \\ 30 \end{bmatrix}$$

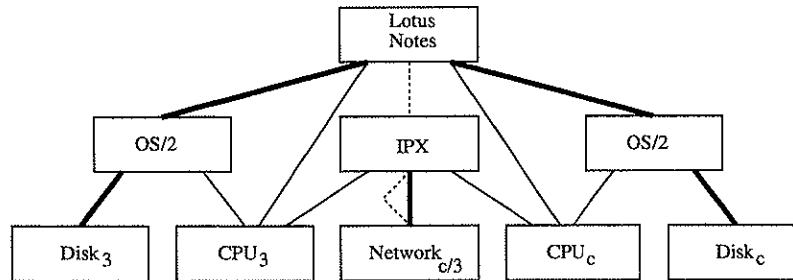
11.6. Evaluate and Validate Static model for Computerised Solution 187

In the parameterised model, some parts of the system in Figure 11.8 were not considered. The network and the client were omitted in the model as shown in the parameterised model in Figure 11.9. If the omitted parts of the model, for example replication in Section 11.3, were parameterised, the predicted response time would increase. Since the largest response time in the computerised solution, 2.3 minutes, is most sensitive to variations in *Secretary_desk_comp*, it is nevertheless *reasonable* to assume performance requirement P_1^G to be satisfied. Contention effects, i.e. when resource demands are replaced by response times, *may* destroy performance requirement P_1^G . Only further investigation of the omitted parts of the static model and a more detailed contention model can give a better answer. This is further work.

Under expected future workload, performance requirement P_2^G must hold. Since this performance requirement is stricter, more information about the omitted parts of the model is necessary before it is possible to say whether performance requirement P_2^G holds or not.



a) Manual telex solution



b) Computerised telex solution

Figure 11.10: Comparing the (a) old manual solution with (b) the new computerised solution.

11.7 Similarities Between Manual and Computerised Solutions

Figure 11.10 compares the manual with the computerised solution. To focus on the essential difficulties, Figure 11.8 has been simplified as shown in Figure 11.10. Compared to Figure 11.8, Figure 11.10 does not show the customised part of Lotus Notes in Figure 11.6 and the human part of the computerised solution in Figure 11.7.

Although the manual and the computerised solutions are fairly equal there are some differences:

- The server (i.e. the module OS/2₃, Disk₃ and CPU₃) is almost missing in the manual solution. This is only imaginary since it is similar to the archives in Statoil in Figure 10.3.
- The secretary performs some of the communication herself, namely the communication which is internal to the sales telex administration sector. The mail man is used for external communication in the manual solution. In the computerised solution, all communication is performed with the (complex) network.
- `Secretary_bus` has a parallel in the bus in between Disk₃ and CPU₃, which is not modelled explicitly in Figure 11.10 (b), and the local bus in the `Networkc/3`.
- `Secretary_store` is an (active) interface to storage. This is the same situation as for the disk, which may seem to be the physical disk, but which in reality also is the interface.

The operations `walk`, `split`, `log`, `stamp`, `copy` have no manual operation, and are almost completely computerised. For the other operations, the manual operation is commented on below:

Find_int: Stored in the name and address book. However, the telex secretary still has to find the correct distribution lists and user profile in Lotus Notes.

Find_ext: Same comment as for Find_int.

Read: Most of this is still manual, but the time consuming job of handling incoming receipts are no longer necessary.

Type: Because of cut and paste, less typing is necessary.

The saving with the computerised solution has to do with using cut and paste of incoming telexes from external agents. Both in the manual and the computerised solution, old archived telexes produced internally can be reused. In addition, some

extra work must be performed in the computerised solution, e.g. understanding the user interface and moving the cursor.

BPR principles could be used to improve the performance even more. Some of the secretary processes may for example be taken over by the contract specialists.

When only one secretary is using this application, no load-dependent effects need to be taken into consideration, except for the other applications using the same resources. For scaling up the case study, load dependent effects have to be investigated.

11.8 Generalisation of the Case Study

The aim of this research is to investigate performance engineering of workflow systems. In this section we generalise the findings of the Gas Sales Telex Administration Case Study. If we compare the resource model in Figure 10.7 with the process for outgoing telexes in Figure 10.6, we realise that the `Telex_secretary` module in Figure 10.7 provides resources to the six subprocesses in Figure 10.6. Each of the six *subprocesses* in Figure 10.6 uses the three modules `Secretary_comms`, `Secretary_proc` and `Secretary_memory` in Figure 10.7. This is illustrated in Figure 11.11.

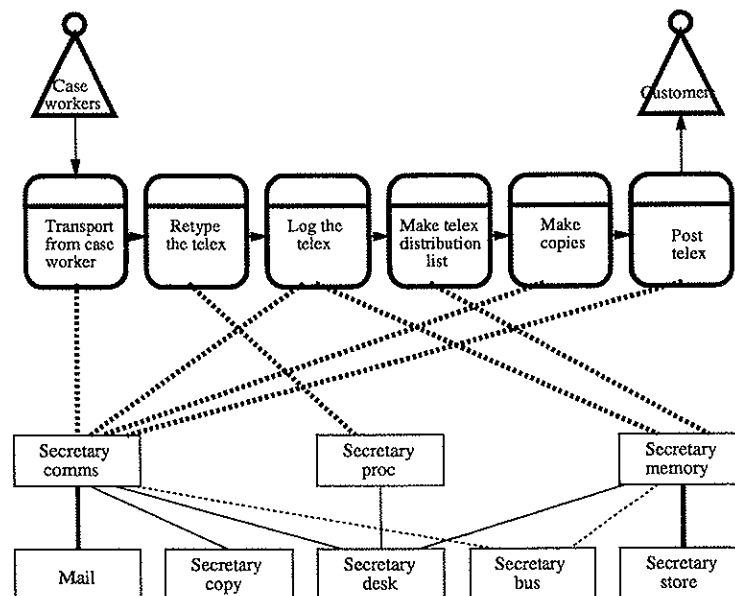


Figure 11.11: Connecting the process model with the resource model in the manual solution. The link between the PrM process model and the SP resource model are indicated with dashed lines.

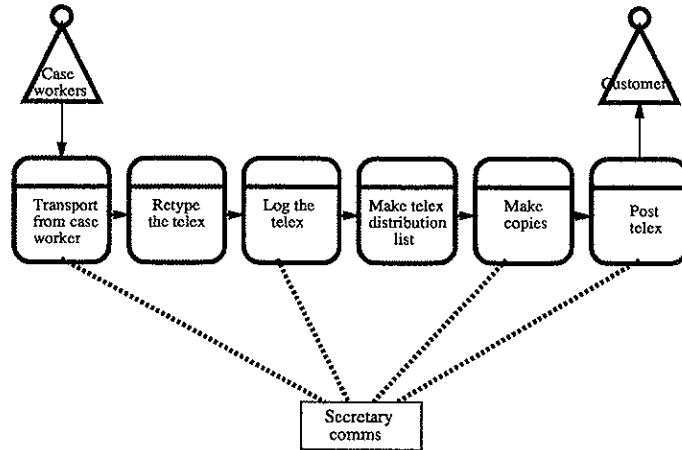


Figure 11.12: Only four subprocesses needed communication. Communication was performed by the SP resource `Secretary_comms`.

The resources `Secretary_comms`, `Secretary_proc` and `Secretary_memory` in Figure 11.11 represent specialised operations for the case worker with the specialised task *telex secretary*. These specialised operations may be possible to generalise, which is an area of further research.

The dashed links in Figure 11.11 show the same complexity specifications as the complexity specifications between the module `Telex_secretary` and the modules `Secretary_comms`, `Secretary_proc` and `Secretary_memory` in Figure 10.7. These complexity specifications are termed $C_{Secretary_comms}^{Telex_secretary}$, $C_{Secretary_proc}^{Telex_secretary}$ and $C_{Secretary_memory}^{Telex_secretary}$ and were described in Section 10.2.1.

In particular, the complexity specification $C_{Secretary_comms}^{Telex_secretary}$ can be identified in Figure 11.11 as the dashed links to the module `Secretary_comms`, i.e. the links from the subprocesses `Transport for case worker`, `Log the telex`, `Make a telex distribution list` and `Post telex`. This is illustrated in Figure 11.12. The complexity specification $C_{Secretary_comms}^{Telex_secretary}$ is also shown below:

$$C_{Secretary_comms}^{Telex_secretary} = \begin{matrix} in_telex \\ out_telex \end{matrix} \begin{matrix} & send_draft & send_internal & fetch_mail & send_mail & fetch_receipt \\ \left[\begin{array}{cccccc} 0 & 1 & 1 & 0 & 0 \\ x & 1 & 0 & 1 & 1 \end{array} \right] \end{matrix}$$

For the computerised solution, the six *subprocesses* in Figure 10.7 use the modules `Customised_Lotus_Notes` and `Secretary_desk_comp` as illustrated in Figure 11.13. Since each subprocess will use both the `Secretary_desk_comp` resource and the `Customised_Lotus_Notes` resource, links between these models are not shown in the figure. As discussed in Section 11.7 the computerised solution performs much

the same tasks as the manual solution, e.g. communication in the computerised solution is handled by the network. If the process in Figure 10.6 had been designed with the computerised solution in mind it might have been revised, e.g. the **Transport_from_case_worker** process could for example be ignored since it is now taken care of by the network, and is no longer an explicit part of the solution.

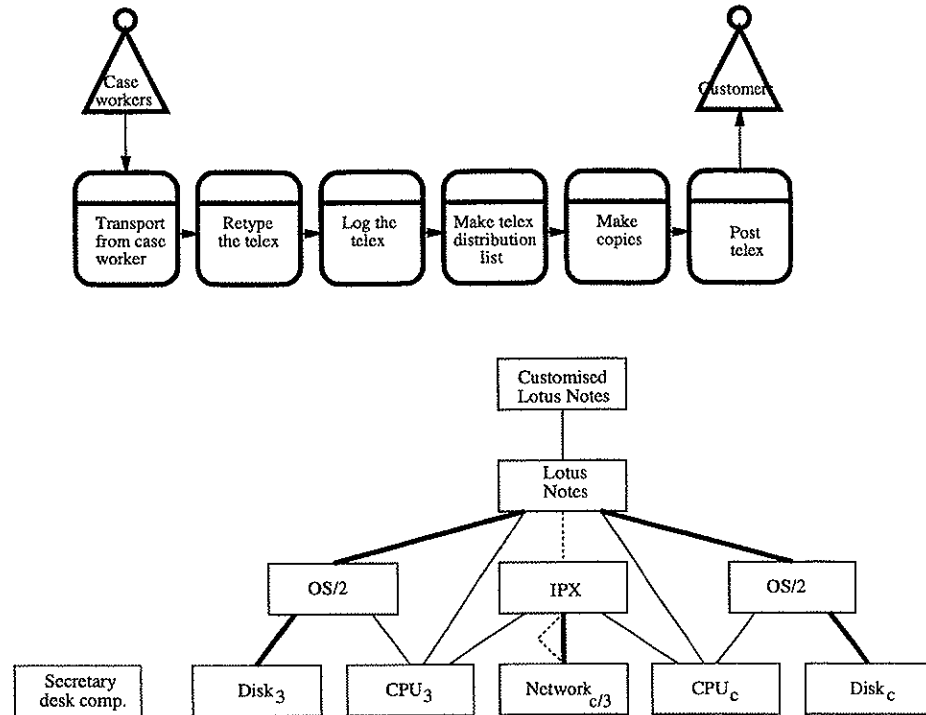


Figure 11.13: Connecting the process model with the resource model in the computerised solution. The links between the PrM subprocess model and the two SP resources `Secretary_desk_comp` and `Customised_Lotus_Notes` are not shown in the figure. All the six subprocesses connect to each of the two resources.

11.9 Summary of Findings

This chapter has given an outline of a general model of the client-server part of Lotus Notes. A work model of the replication in Lotus Notes has been outlined. Similarities in the work model of the computerised and in the manual solution were identified. There is no overall validation in this chapter, because it is not possible to validate a system which is under development.

Model structure is coarser here than in Chapter 10. This is because availability of parameters and the amount of work involved made it harder to make a detailed

model. This illustrates two granularities for modelling human workflows.

All the interconnections between the several parts of the method resulted in a number of iterative refinements of the models in this case study. This is in accordance with the method outlined in Chapter 6.

In this case study, the number of details was quite high. Since several worlds also were involved, the concept of several local realities was noticeable. Each local reality would roughly correspond to one component in the SP model. If time had permitted, it would have been possible to make a more detailed model involving a larger number of components (c.f. Figure 11.8). It would also been possible to go further into the different worlds involved. Making such a large model would require the support of a tool integrating PPP and SP.

In the ideal case, general models presented in this chapter should have been supplied by the software vendor, in this case the Lotus Notes vendor. These general models would then serve as work model platforms and make work modelling during development easier.

Chapter 12

Conclusions and Further Work

This chapter sums up the perceived achievements of this work and points out aspects which should be dealt with more extensively, giving directions for further work.

12.1 Major Contributions

The major achievements for each part of the contribution are described below.

A method for performance engineering of information systems is described, based on the SP method. This is a step forward with respect to the state of the art in the field of performance engineering of information systems.

As described in Section 5.3, the primary motivation for performance engineering of information systems is to decrease the cost and risk of system development. When performance engineering successfully augments information system development, timely completed information systems which satisfy performance requirements can be realised. Even though systematic performance engineering is not common practice in information system development, current *best* practice shows that performance engineering of software is possible, and cost effective for some systems, e.g. the SPE method [Smi90]. The method in this thesis represents an improvement of the state of the art in the following respects:

Design description Easier interaction between the design description and the performance model. This is demonstrated by annotating PrM models with performance parameters, for example in the Blood Bank Case Study. The resource concept in SP complements the process models and data models of common CASE-tools, in this case the experimental CASE-tool PPP [GLW91]. Work budgets are also more convenient during development than budgets in terms of performance (c.f. Section 5.4.6).

World concept In SP the world concept is made explicit, deepening the understanding of performance for large systems. The method is based on this "world" concept. Two worlds of competence are needed in performance engineering. Information system developers know details about the developed application (but have often only superficial overview of the hardware and software platform), and system architects have an overview of the total system (including hardware and software platform, but often only superficial overview of the application). Information system developers, with low performance engineering competence, but with high information system competence, could assist in doing performance engineering in the early phases of systems development. In particular, such personnel could be instrumental in resource annotation before professional performance engineers (e.g. systems architects) eventually take over and detail the performance models based on the most important parameters.

Hierarchies In SP it is easier to build hierarchies at a meaningful level of abstraction (c.f. Section 6.2.1). Explicit platforms in SP is an example of a very useful hierarchy level. Together with the hierarchies, the typing in SP (c.f. Section 5.4.6) makes reuse more likely [Vet93].

Organisations It is quite easy to extend the PPP/SP method to organisations as described in this thesis. For an information system, the focus on the organisation is more important than for a software system. Therefore, this is an information system performance engineering method and not only a software performance engineering method.

The method is explored and illustrated with two case studies, the first focusing on a transaction-oriented information system, and the second on workflow systems and the organisation around a workflow system. The number of parameters in both case studies was not too great. Both in the Blood Bank Case Study and in the Gas Sales Telex Administration Case Study the number of necessary parameters was in the order of 100. The number of necessary parameters may be reduced after one iteration of parameter capture and model projection. In both case studies it was possible to make the model as accurate as needed.

An overall resource framework which integrates both human and computerised processes and resources in the same framework has been developed.

An integrated approach to performance analysis of workflows in terms of throughput and response time to satisfy customer requirements is needed [VLP95]. The developed framework is promising and represents a path for further work. The framework should make it easier to handle performance issues in workflow systems in a more coherent way, as described in the paper [BSH94]. When humans and computer resources are considered in the same system, tradeoffs between them become clearer.

For information processing, there is no fundamental difference between humans and computer resources. Human and computerised processes can be modelled in the

same framework, if they belong to routine or template processes. More research efforts are needed for ad-hoc workflows.

Some SP concepts are more readily understood after the introduction of human resources, e.g. communication as memory with low persistence (see Section 4.7.).

12.2 Limitations and Further Work

For each part of the contribution, issues should be addressed in future work. For performance engineering, these issues need more consideration:

- The SP tools should be integrated with the PPP tools. A fully integrated tool should be built. Whereas the coupling between PrM and SP was implemented by Opdahl [Opd92], the coupling between PhM and the data modelling in SP needs more consideration.

More flexibility is also needed in the SP tool if it shall be helpful also in early phases of system development. It should be possible to specify ADTs without having to specify all the modules down to primary modules.

- Boundaries of the discussion of the method was described in the beginning of Chapter 6. It should be investigated if aspects which have not received thorough attention in the discussion calls for refinements in the method.

Concerning the integrating of human and computerised processes and resources, issues for future work are:

- The SP model shown in Figure 10.7 may be the beginning of a more general model of case workers, because communication, processing and storage is the only way of handling information. In a more general SP model, the processing operations will be more complex and need more research. The typing rules in SP need more investigation in the organisational world.
- Administration of Lotus Notes performance may be problematic [Bru95]. Further work on the Lotus Notes modelling in this thesis could provide part of a solution to this problem. The same ideas should also be applicable to performance problems in WWW servers.
- According to the FRISCO group [FRI95], little work has been done to assess the effectiveness of information systems. Integration of this framework with frameworks for other effectiveness criteria, both for information systems and for organisations, may be one path to follow.
- Performance of the computerised information system will often set the boundaries for the solutions which may be selected at the organisational level [DS90],

and it is therefore important with an interaction between performance engineering and BPR. Tools for monitoring performance are especially important for workflows crossing organisational boundaries [JLP⁺95]. This framework needs to be extended more thoroughly into the organisation. A closer look on BPR and the principles governing BPR may be necessary.

To do BPR, one must know something about the intentions as described by Yu [Yu94]. Krogstie has also modelled goals in the PPP framework [Kro95]. Yu's thesis models goal and softgoal relationships in addition to resource and task relationships. It would be interesting to integrate these relationships into the PPP/SP framework.

Appendices

Appendix A

Replication in Statoil

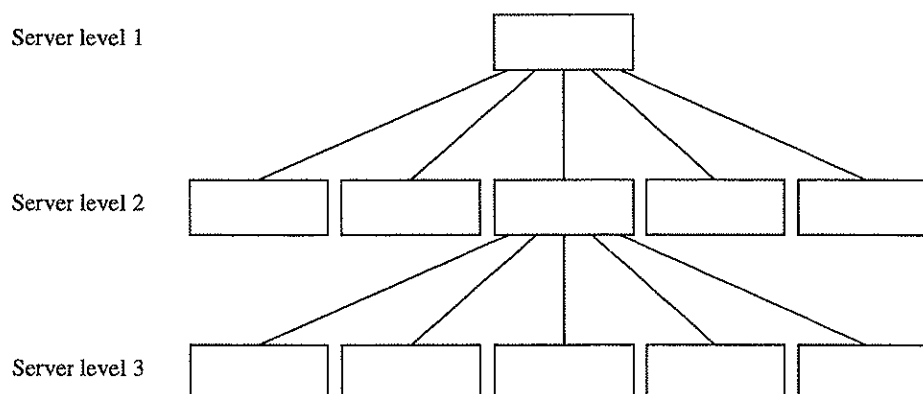


Figure A.1: Three server levels in Statoil

This chapter explains the replication used in Statoil at an overall level. Ineffective replication scheduling is an often-underrated cause of suboptimal performance for Lotus Notes [Yav94]. Point-to-point replication with every server would have been an easy, but also a resource consuming way of doing replication. Instead, Statoil uses hierarchic replication with three server levels. As depicted in Figure A.1, each server at server level 1 has servers at server level 2 as nodes, and servers at server level 2 have nodes at server level 3.¹

A.1 Replication Plan

The replication plan in Statoil consists of three activities. The first and third activity are hub-and-spoke replication and the second activity is point-to-point replication.

¹This figure is not an SP model. An SP model represents logical structure of work, whereas this model represents physical server structure.

Hub-and-spoke replication is a structured, stable, and simple-to-maintain means of performing replication [Sta93]. With hub-and-spoke replication, the hub initiates replication with each node. A Notes hub is a server dedicated to routing mail and replication databases to other Notes servers [Lot93a]. The servers to which a hub connects are designated as “spokes” [Lot93a].

Activities 1 and 2 in the replication plan are performed in parallel, followed by activity 3 [Sta93]:

Activity 1 Server level 2 initialises replication with servers at server level 3, for one server at server level 3 after the other. After the last server node is replicated, the whole process is repeated but now in the opposite direction. This ensures that changes in the servers at level 3 are distributed to all the other servers at server level 3 with the same server level 2 hub.

Activity 2 All the servers at server level 1 are replicated with each other in a chain, one by one. Finally, the last server in the chain is replicated with the first server, which distributes all the changes between all servers at server level 1.

Activity 3 Server level 1 is replicated with server level 2 in the same way as in activity 1 above.

When activity 3 is finished, activities 1 and 2 start in an endless loop. Activity 1 and activity 2 each takes 3 hours, including 0.5 hours for extra point-to-point replications. Since the maximum number of nodes for each hub is 10, this leaves 15 minutes for each node to replicate in both directions ($10 \cdot 15 \text{ minutes} = 150 \text{ minutes} = 2.5 \text{ hours}$).

Since each activity takes 3 hours, and activities 1 and 2 are performed in parallel, the complete replication plan with activity 1, 2, and 3 takes 6 hours. When this replication plan is repeated 2.5 times (i.e. activity (1 and 2), 3, (1 and 2), 3, and, (1 and 2)), changes in every server is propagated to all the other servers in the network. Hence, it will take maximum 15 hours ($2.5 \cdot 6 = 15$) before changes in one server is distributed to all the other servers.

This simple hub-and-spoke replication may be extended with some other means of replication:

Point-to-point replication As a rule of thumb, if a database is needed in only one or two servers, point-to-point replication should be used [Sta93]. Otherwise, the replication plan sketched above can be used, or the **replication to level 2 only** below.

Replication to server level 2 only If a database is needed only in a few offices, it might be sufficient to replicate it to server level 2 instead of all the way up to server level 1.

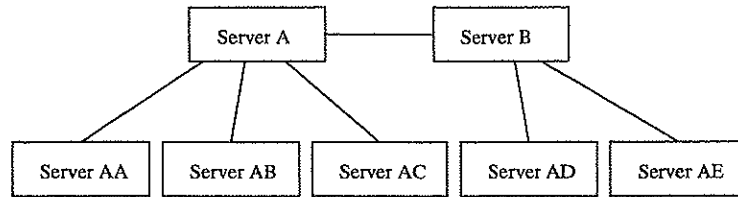


Figure A.2: Crossreplication between two Server A and Server B.

Crossreplication If the load on one server becomes too large, the server can be extended with one more server as depicted in Figure A.2 where Server A is extended with Server B. Thus, the load on Server A is reduced, because Server B takes responsibility for Server AD and Server AE which previously belonged to server A.

To determine the order of replication, three priorities are used [Sta93]:

Low Used for databases which should be replicated to every server at server level 1.

Medium For crossreplication at server level 2.

High Not used now, but might be used for crossreplication at server level 3.

Server connections for replication and remote mail routing are determined by connection documents in the name and address book. Network connection documents are for servers in the same network and the same domain and remote connection documents are for servers in different networks or different domains [Lot93a].

A.2 Network Service Classes

The capacity of networks differ, so Statoil distinguishes between three service levels [Sta93]:

Service level 1 Full WAN (wide area networks) connection with 64 kbps or higher capacity (1 kbps = 1000 bits per second). Since LANs (local area networks) offer higher capacity than WANs, LANs are also included here. Typical capacities for LANs and WANs respectively are:

LAN maximum capacity in the 10 Mbps area. More specifically, Ethernet has a maximum capacity of 10 Mbps, the Token ring has a maximum capacity 4 Mbps and 16 Mbps and FDDI has a maximum capacity of 100 Mbps.

WAN The national telecommunication agency delivers capacity in 64 kbps increments, e.g. 256 kbps and 384 kbps.

Service level 2 Full WAN connection, with less than 64 kbps capacity.

Service level 3 No WAN connection, with less than 64 kbps capacity. Typically this is a dial-up communication, e.g. 9600 bps to Warsaw.

With service level 2, it is useful to have more local databases than with service level 1. All databases must reside locally with service level 3.

A.3 Email Transport

Emails in Statoil need faster transport than replicated databases. If the receiver server at server level 3 is a node of the same server level 2 hub as the sender server at server level 3, point-to-point routing between these servers is used. But if the receiver server is controlled by another server level 2 hub, three point-to-point routings are used to transport the email [Sta93]:

Routing 1 From the server level 3 sender server to the server level 2 hub.

Routing 2 Between the sender hub at server level 2 and the receiver hub at server level 2.

Routing 3 From the receiver hub at server level 2 to the receiver server at server level 3.

This approach is depicted in Figure A.3. No hub needs to take care of other emails than the emails which are local to this hub, thereby reducing the server workload and simplifying cost estimation.

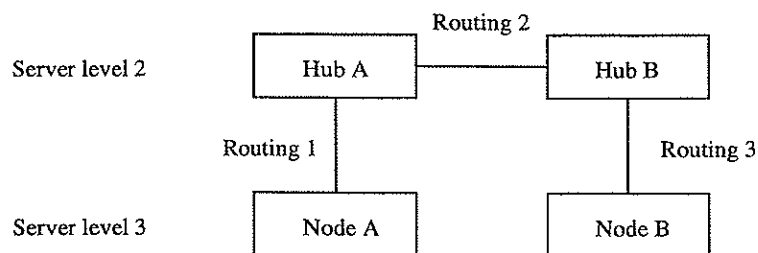


Figure A.3: Email transport between two server level 3 server nodes with different server level 2 hubs, consisting of three routings.

Appendix B

The Experimental CASE Tool PPP

To show how these solutions can be exploited for the PPP (Phenomena, Process, Program) approach we will specify how they can be included in the PPP CASE tool. The revised version of PPP is under development and thus, the features suggested in this work are not yet implemented.

The PPP approach comprises a set of languages, methods and tools to support information systems development, from initial problem description and requirements specification to implementation and maintenance. This section presents the PPP languages and the implementation of the PPP CASE tool.

This chapter is more or less directly copied from Seltveit's thesis [Sel94] with her permission.

B.1 PPP Languages

PPP languages are: Phenomenon Language (PhM) [Sø79, Sø80, Opd88, Yan93], Process Language (PrM) [BC86, Opd88, GLW91] and Process Life Description (PLD) [Wil93]. The basic constructs of the PPP languages are adopted from well-known and widely used modelling languages. PhM is an extended ER language and describes the static aspects of the UoD. PrM and PLD describe the dynamic aspects and are based on Structured Analysis [GS77, deM78] and graphical program specification [Tri88], respectively.

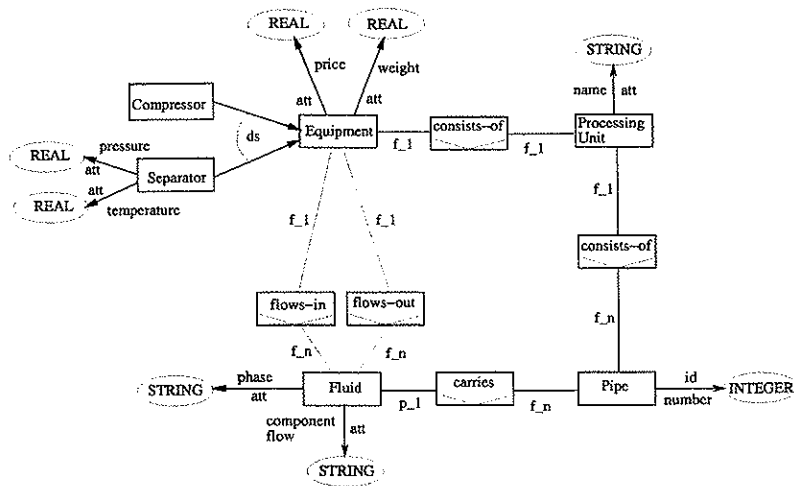


Figure B.1: An example of a PhM model.

B.1.1 PhM

PhM (Phenomenon Modelling Language) is an extension of the traditional Entity-Relationship language. The basic modelling constructs of PhM are: entity classes, relationship classes, attribute relations and data types. PhM distinguishes between data entities and non-data entities. The non-data entities describe real world objects whereas the data entities describe the informational counterparts, i.e., information about the real world objects. An example of a PhM model is shown in Figure B.1 and as we see, the graphical notation is slightly different from traditional ER notation.

Entity classes, relationship classes and cardinality constraints have the same meaning as in the ER. To increase the expressive power, the PhM contains the following extensions compared to the traditional ER: attribute relations, relationship constraints, generalisation/specialisation hierarchies, conceptual views and a PhM algebra. For more comprehensive descriptions, the interested reader is referred to [Sø77a, Sø79, Sø80, Opd88, Yan93].

An **attribute relation** links a data type to an entity. Attribute relations are classified into four types [SK93]:

- *Identifier* which uniquely determines instances of an entity class (denoted *id*).
- *Attribute* which is used to denote single-value properties to instances of classes (denoted *att*).
- *Repeating group* which is used to denote multiple-value properties to instances of classes (denoted *rep*).
- *Quality* which is used to denote properties that are characteristic of a class rather than of the elements of the class (denoted *qual*).

A **relationship constraints** specifies if an entity class participates fully or partially in a relationship. An entity class relates fully (denoted *f*) in a relationship if all the instances of the entity class participate in the relationship; otherwise it relates partially (denoted *p*) in the relationship.

Generalisation is supported through the subclass concept (denoted *S*). An entity class may have several subclasses. All the members of a subclass are also members of the subclass' superclass and they inherit all the properties of the superclass.

Conceptual views¹ are introduced to allow definition of different *views* of a domain, i.e., each conceptual view consists of a collection of entity classes, relationship classes, *etc.* They can be used in a similar manner as how SQL views [Dat86] are used to define various views of an SQL application. A PhM model can thus be extended or modified by adding new conceptual views. By providing an algebra, extensions of the model can be done by defining new components, e.g., entity classes, in terms of already defined components.

The **ONE-R algebra** is developed by M. Yang [Yan93]. It is based on relational algebra but it is extended to manipulate components in a PhM model. The ONE-R algebra provides the following operators: union, difference, product, selection, projection, reduction, nest, unnest, pack, unpack, join, natural join, relationship join and packed relationship join.

B.1.2 PrM

PrM (Process Modelling Language) is based on the process language of the PPP language, and is an extension of regular DFDs [GS77]. It specifies processes and their interaction in a formal way. The interaction between processes is modelled using the concept of flows. PrM specifications include the interactions between processes at the same level of abstraction, as well as how processes at any level of abstraction relate to their parent process and their decompositions. The basic modelling concepts of PrM are: processes, data stores, external agents, flows, ports, buffers, items, resources and timers. An example of a PrM model is shown in Figures B.2 and B.3.

The concepts of process, external agent, data store, and data flow have the same meaning as in the traditional DFD. The external agent in PrM corresponds to the source/sink concept (i.e., external entity) in DFD. To increase the expressive power, PrM contains some extensions compared to DFDs. For more detailed descriptions of PrM, see [BC86, Opd88, GLW91].

A **flow** is modelled as a channel carrying *items*. Items are the objects stored in stores, sent or received by external agents and carried by flows. The concept of flow

¹A conceptual view was initially called a scenario [Sø77b].

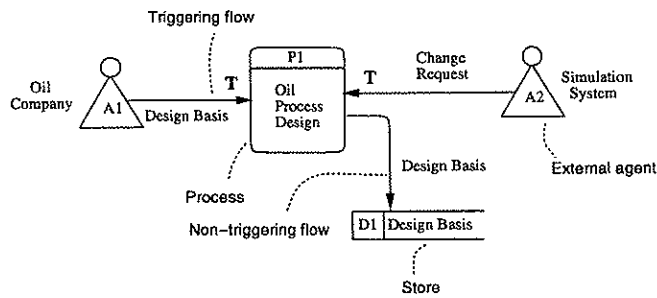


Figure B.2: Top-level PrM model for oil processing design.

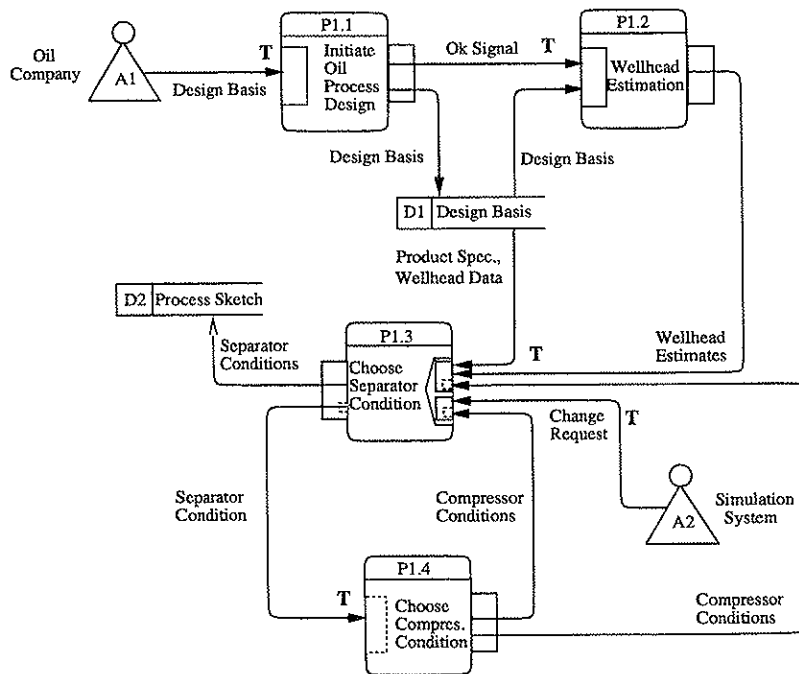


Figure B.3: Decomposed PrM model for oil processing design including ports.

is extended to denote *material flow* as well as *dataflow*.² No distinction is made between flows carrying data and flows carrying material. Furthermore, flow may be *triggering* or *non-triggering*. Triggering flows determine when a process should be considered for firing. A process is triggered when the items on its triggering flows have arrived. Triggering of a process implies that a process *instance* is created. A trigger may be triggered one or more times and each time a new process instance is created. Triggering flows are depicted in the diagrams by a **T** in the processes where the flow ends. Triggering and non-triggering items may be *buffered* upon arrival at a process class if there is no process instance to consume them. Such items are stored in a **buffer**. Items, buffers and resources are not depicted in a diagram. *Terminating flows* indicate the termination of a process' execution, and are depicted in the diagrams by a **T** in the processes where the flow starts.

A *resource* contains items that are necessary for a process to run. A resource may be connected to several processes and items are retrieved and returned by the processes. Resources are diagrammatically depicted by a circle with the resource name written inside.

The concept of **store** is extended to denote a *material* store as well as a *data* store. Stores denote abstractions of repositories of information and material.

A **port** is a graphical formalism for expressing logical relations between input flows (and output flows, respectively) of processes. A process can have one or more input ports and one or more output ports. Ports define the set of valid combinations of input flows and output flows which will make up a *process instance*. Ports can be classified into three types (the graphical notation is depicted in Figure B.4):

- *Plain ports* (AND, OR and XOR) depict that the items on the flows enter the process once only, i.e., in one 'batch'. The meaning of the ports:
 - An AND port depicts a logical AND relation between the flows.
 - An OR port depicts a logical OR relation between the flows.
 - An XOR port depicts a logical XOR relation between the flows.
- *Repeating ports* (REP) depict that the items on the flows enter the process one or more times, i.e., in one or more 'batches'.
- *Conditional ports* (COND) depict that the items on the flows may enter the process once, but only if the condition that applies is satisfied.

The different ports can be used isolated or nested to form composite ports. An example of the former is shown in Figure B.5. The process has 4 possibilities: 'arrivals on a', 'arrivals on b', 'departures on c' and 'departures on d'. The process has two possible (valid) input combinations, either 'arrival on a' or 'arrival on b' but only one valid output possibility, namely 'arrival on c and d'.

²Real-world modelling in PPP is still a research topic. Decomposition of material flows is for example investigated in [OS94].

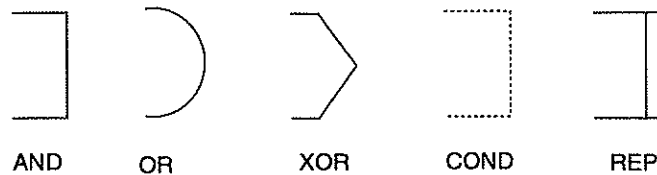


Figure B.4: Port symbols of PrM.

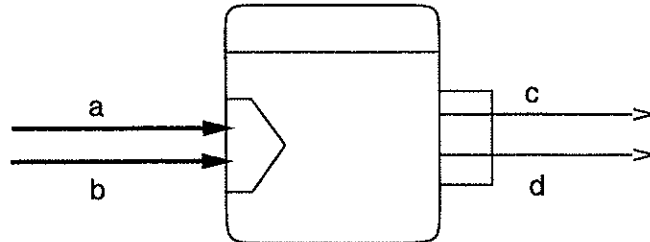


Figure B.5: A simple port example.

An example of how composite ports can be used is shown in Figure B.6. The process has 7 possibilities: 'arrivals on a', 'arrivals on b', 'arrivals on c', 'arrivals on d', 'departures on e', 'departures on f' and 'departures on g'. There are three possible input combinations: (b) (i.e., one b flow), (a, b) (i.e., one a flow, one b flow) and (c, d) and three possible output combinations: (e, f, g), (e, g) and (f, g). This yields a total of 9 acceptable flow combinations for the process as a whole.

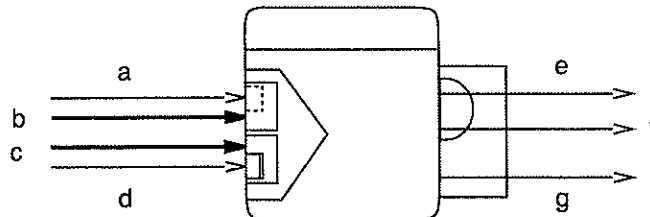


Figure B.6: A small composite port example.

Timers are either clocks or delays. Clocks are used to model events that are to occur at a specific moment in time. Delays are used to model events that are delayed a certain time interval and are specified relative to the occurrence of a flow.

In addition to PrM, an overview facility called **Process Hierarchy (PH)** is also provided. A PH is introduced to represent hierarchical relationships between processes and their decompositions. No ordering is implied in the diagram. Each non-leaf node of a PH is described by a PrM. Figure B.7 shows the PH generated from the PrMs of Figures B.2 and B.3. Processes on higher levels of abstraction are pictured 'higher up' in the Process Hierarchy. It should be noted that each process in PrM is allowed to be represented by only one icon in the PH.

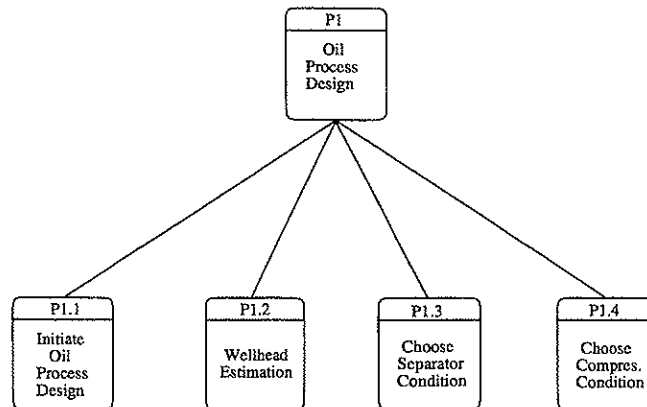


Figure B.7: An example Process Hierarchy.

A complete PH for a given instance of a process model shows all the processes in the corresponding PrMs, i.e., all processes at all levels of decomposition. For some applications, the number of processes may grow large and it may appear useful to split the PH into several smaller PHs, each depicting a limited part of the decomposition hierarchy.

B.1.3 PLD

PLD (Process Life Description) is a procedural language and it is used to describe the process logic of non-decomposed processes. Constructs of PLD are:

- Start: indicate beginning of a PLD specification
- Receive: receive dataflow
- Send: send dataflow
- Assignment: symbolise a block of program statements or a subprogram call
- Selection: IF-test
- Loops: FOR and WHILE-loops

An example of a PLD specification of process P1.3 `Choose_Separator_Condition` in Figure B.3 is shown in Figure B.8. When the ports are designed in a PPP diagram, the skeleton of PLD is generated automatically, taking care of the reception and sending of data. Further design details can then be added. For a detailed description of PLD, see [Wil93].

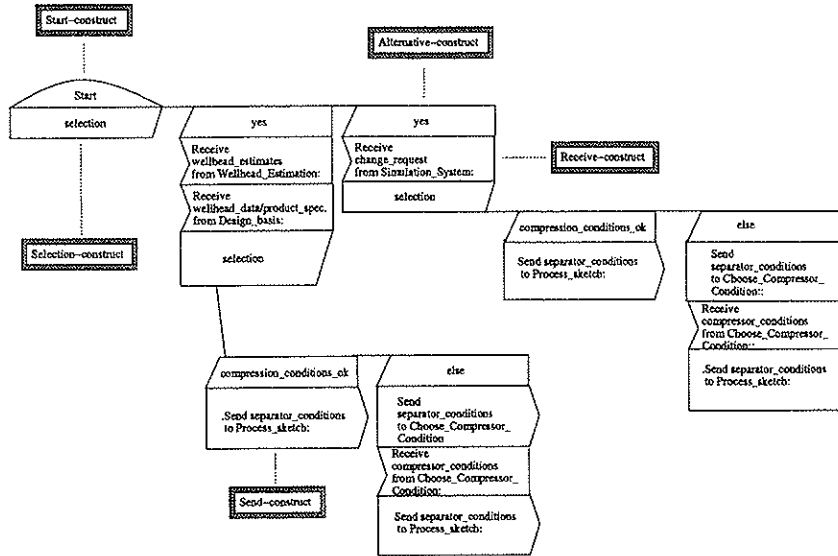


Figure B.8: An example of a PLD specification.

B.2 PPP CASE Tool

A revised version of a prototype CASE tool environment for PPP is under development. Compared with the previous version [GLW91], new features such as explanation generation, performance evaluation, and complexity reduction will be provided. A graphical representation of the current PPP implementation is presented in Figure B.9, the various components of which we will detail by first describing the purpose of each module and secondly how each module is implemented.

B.2.1 Purpose of Modules

- **Modelling editors** serve to allow the creation and checking of the conceptual and design level models of the application being developed. The editors facilitate creation, retrieval, modification, deletion and storage of models represented in PhM, PrM and PLD.
- **PPP repository** holds a copy of the conceptual and design models in a database, allowing access to the information through a versioning facility.
- **Analysis support** provides verification and validation support. Verification includes syntax checking and checking of semantics of PPP models. Validation includes translation of PPP models to executable prototypes.
- **Explanation facility** provides means to explain phenomena represented in conceptual models. The explanation generation facility for PPP offers the user

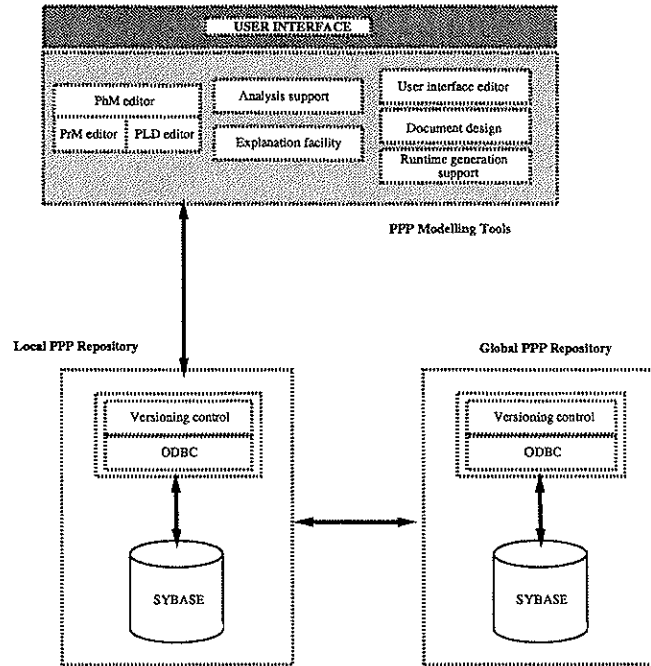


Figure B.9: Implementation architecture of the PPP CASE tool.

a set of questions to be asked to increase their understanding. It can also take into account user characteristics and the context in which the questions are asked.

- **User interface editor** serves to allow the creation and checking of static and dynamic aspects of user interfaces of the application being developed. It is based on the features offered by graphical user interface technology.
- **Document design** serves to allow the creation and storage of documents with specific formats, e.g., reports, forms and sheets.
- **Runtime generation support** provides support for the generation of the run time system as well as for the execution of systems developed using the PPP approach.
- **Sybase** is the RDBMS used by any application built by the PPP tools when it is executed on the PPP runtime system.

B.2.2 Implementation of the Modules

The prototype is being developed on SUN Sparc workstations which run UNIX Solaris 2.3. C++ is used as the implementation language for most of the functional

features. Exceptions are some of the modules which are already implemented separately in BIM-Prolog and are included in the new version of the PPP tool by using the C++/Prolog interface. The user interface is built using the commercial user interface packages Interviews and Unidraw. All the services provided by the various modules described below are accessed through this common user interface. The following list gives a brief description of the implementation of each module. The communication between modules is made using the local PPP repository.

- **Modelling editors** including the **document design editor** are built using Interviews, Unidraw, and C++. Interviews and Unidraw provide the graphical primitives necessary for building a graphical and form based CASE tool and is coupled with the PPP repository which acts as the persistent storage mechanism for the tools when in use.
- **PPP repository** holds a copy of the conceptual and design models in a database, allowing access to the information through a versioning facility. The repository also provides the means for integrating the various PPP tools through data sharing. The storage structure (also called the PPP repository structure) is partly³ derived from the PPP meta-model.
- **Analysis support** provides verification and validation support. Verification is to be provided as an integral part of the modelling editors. The user can choose the target language for the generation of executable prototypes.
- **Explanation facility** is implemented using Bim-Prolog.
- **User interface editor** is implemented using Interviews and Unidraw.
- **Sybase** is a commercial RDBMS which is used without modification as the RDBMS in the PPP (for both development as well as for the runtime system).

³The PPP repository structure must also convey some additional information about administrative matters, *etc.*

List of Figures

1.1	Schematic relation between three performance engineering theses in the Information Systems Group.	2
2.1	Top level blood bank model. These four information subsystems can be decomposed into 34 processes [Flø91].	6
2.2	The manual transfusion process in the blood bank organisation.	7
2.3	The transfusion process in the blood bank as it was implemented. Manual processes are indicated by boldface process symbols.	8
2.4	Basic framework for performance engineering in workflow organisations. The legend is in Figure 2.5.	10
2.5	Legend for the basic framework in Figure 2.4.	11
2.6	The structure of this thesis in terms of Figure 2.4.	11
2.7	A sketch of the resources needed for earlier prediction of potential problems. The legend is in Figure 2.5.	12
3.1	W gives workload on the system, S , with the resulting performance P	15
3.2	Both the small system and the large system have a workload. The workload on the small system belongs to the large system.	16
3.3	Two feedback loops causing problems for workload characterisation.	17
3.4	An “information system” which does not take (hardware) resources into account in contrast to a “holistic” performance system where all the resources are included in the system.	18
3.5	An example of a simple open queueing network. This queueing network represents a CPU and three disks as a queue centres.	20
3.6	A graph of the relation $R = \frac{D}{1-U}$	21
3.7	A simple Petri net.	22
3.8	The performance modelling cycle [LZGS84].	24
4.1	Type of resources used by an application. At the bottom, the resources are physical and concrete like memory, CPU or disk. Higher in the resource hierarchy, the resources are abstract. Not visible in this figure is the organisation which uses the application.	26
4.2	SP model for a secretary.	27
4.3	Framework for performance engineering of computerised information systems. The legend for this framework is shown in Figure 2.5 in Section 2.2.	28
4.4	Work is always related to the use of operations at a given abstraction level.	28

4.5	Combining a work model, a load model and a contention model gives a performance model [Vet93].	29
4.6	Work complexity is a matrix showing the relation between two levels of operation.	30
4.7	Each <i>class</i> represents one way of using the <i>operations</i> of a <i>system</i> . . .	33
4.8	Hierarchical performance modelling [Hug95, p. 40].	36
4.9	The virtual machine processes information with some operations. . .	37
4.10	The outer process invokes an operation on the virtual machine and an inner process is started.	38
4.11	An abstract data type and a data model.	38
4.12	A module specification with data models.	38
4.13	SP model for a secretary. Links between components are typed. Processing operations have solid lines, memory operations have bold lines and communication operations have dotted lines.	41
4.14	Discrimination in SP.	43
4.15	The non-distributed module A uses the submodules B for storage and the submodule C for processing.	43
4.16	The module A is distributed.	43
4.17	A client-server architecture.	43
4.18	A variant of the client-server architecture.	43
5.1	The waterfall model of the software lifecycle [Boe81].	49
5.2	The complementary functional and performance prototypes may be driven by a prototype interactor.	51
5.3	The spiral model [Boe88].	52
5.4	The PPP method [GLW91].	54
5.5	Software execution models are used to model software, while system execution models model hardware.	57
5.6	The SPE method [Smi90].	59
5.7	Example of a software process architecture [RS95].	61
5.8	The overall COMPLEMENT process model (Adapted from [CVT+92]).	62
5.9	The process model must be annotated with workload intensities, branching probabilities and resource demands.	64
6.1	Relation between information system development and performance engineering.	72
6.2	Different roles give different types of parameters as illustrated by the broken lines. Roles are <i>emphasised</i>	77
6.3	In (a) the platform model is inside of the application model, whereas in (b) the platform model is outside of the application model.	79
6.4	An example of a platform model in the Blood Bank Case Study. . . .	80
6.5	Budgets guide the design process. If the budgets predict inadequate performance, this may be cured by refinements in the design description, or relaxed (performance) requirements. The names on the links in this figures are more detailed than to the link names in Figure 6.1, e.g. the link "Guide system development" in Figure 6.1 will at least contain the link "Budgets" in this figure.	81
6.6	Decomposition of a module in the SP language and the DFD language.	83

6.7	There will be several DFD processes within an SP module.	84
7.1	An overview of the workload and resources used by the blood bank application.	93
7.2	PrM output port connectives for the <code>fetch info</code> process	98
7.3	Resources used by the <code>fetch info</code> process.	100
7.4	SP model for the Blood Bank Case Study	101
8.1	Leavitt's diamond for organisational change.	110
8.2	Social construction in an organisation [Gje93, Kro95].	112
8.3	Comparing approaches to organisational change [PR95].	115
8.4	An example of a simple workflow.	119
8.5	Levels of abstraction in operating systems [Den92].	123
8.6	The Input/Process/Output paradigm.	123
8.7	The speech act based paradigm paradigm (Adapted from [MMWFF92]. The workflow loop is copyright of Action Technology.).	124
8.8	The Workflow Reference Model shows how five subsystems interoperate with the Workflow Enactment Service [Coa95].	125
8.9	Basic elements of a form.	126
8.10	A view lists several fields for each form. This view is sorted on the date field.	127
9.1	An organisation must at least consist of the individual level, the group level and the organisational level illustrated in the SP modelling style.	130
9.2	The framework for performance engineering in workflow organisations. The legend for this framework is shown in Figure 2.5 in Section 2.2.	131
9.3	The workflows in the blood bank and in the Gas Sales Telex Administration Case Study differ in degree of determinism.	134
9.4	Three principles of work shifting: (a) delegation, (b) resource switching and (c) empowerment.	136
9.5	Internal architecture of a workflow system.	138
10.1	Distinction between three relevant systems	141
10.2	Contract coordination process.	143
10.3	Rudimentary SP diagram for the Statoil organisation	144
10.4	Rudimentary SP diagram for the sales administration sector.	144
10.5	Process for incoming telexes.	148
10.6	Process for outgoing telexes.	148
10.7	SP model for the present telex handling by the telex secretary.	149
10.8	Rudimentary SP model for the present handling by the contract specialists.	163
11.1	Three server levels in Statoil	171
11.2	Separate parts of Lotus Notes modelling	172
11.3	SP model for Lotus Notes	173
11.4	The SP model measured by Kowalski	174
11.5	Replication architecture for Lotus Notes.	176
11.6	Simplified, customised Lotus Notes model.	177

11.7	SP model which shows how the secretary and Lotus Notes are used.	179
11.8	A "complete" SP model for (a) the manual and (b) the computerised solution.	181
11.9	The model which was parameterised for (a) the manual and (b) the computerised solution.	182
11.10	Comparing the (a) old manual solution with (b) the new computerised solution.	187
11.11	Connecting the process model with the resource model in the manual solution. The link between the PrM process model and the SP resource model are indicated with dashed lines.	189
11.12	Only four subprocesses needed communication. Communication was performed by the SP resource <code>Secretary_comms</code>	190
11.13	Connecting the process model with the resource model in the computerised solution. The links between the PrM subprocess model and the two SP resources <code>Secretary_desk_comp</code> and <code>Customised_Lotus_Notes</code> are not shown in the figure. All the six subprocesses connect to each of the two resources.	191
A.1	Three server levels in Statoil	199
A.2	Crossreplication between two Server A and Server B.	201
A.3	Email transport between two server level 3 server nodes with different server level 2 hubs, consisting of three routings.	202
B.1	An example of a PhM model.	204
B.2	Top-level PrM model for oil processing design.	206
B.3	Decomposed PrM model for oil processing design including ports.	206
B.4	Port symbols of PrM.	208
B.5	A simple port example.	208
B.6	A small composite port example.	208
B.7	An example Process Hierarchy.	209
B.8	An example of a PLD specification.	210
B.9	Implementation architecture of the PPP CASE tool.	211

List of Tables

0.1	A brief outline of the thesis.	vii
5.1	Taxonomy of approaches to performance engineering.	66
6.1	Overall method.	71
6.2	Static model method.	76
10.1	Parameters representing average numbers for the manual solution. . .	157
11.1	Parameter representing average numbers for the Lotus Notes solution.	183

Bibliography

- [AB88] Reda A. Ammar and Taylor L. Booth. Software Optimization Using User Models. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(4):552 – 560, July/August 1988.
- [Abb94] Kenneth R. Abbott. Experiences with Workflow Management: Issues for the Next Generation. In *Proceedings Conference on Computer-Supported Cooperative Work (CSCW '94)*. ACM, October 1994.
- [ABC88] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems*. MIT press, Cambridge, Massachusetts, second edition, 1988.
- [ABS91] Rudolf Andresen, Janis Bubenko jr., and Arne Sølvberg, editors. *Advanced Information Systems Engineering*, Trondheim, Norway, May 1991. Springer-Verlag. 3th International Conference, CAISE '91.
- [AC92] Sandra Ayache and Eric Conquet. Integration of Performance Specification and Evaluation in System Development Life-cycle. Technical report, Matra Marconi Space France, 1992.
- [Ale86] Charles T. Alexander. Performance Engineering: Various Techniques and Tools. In *Proc. CMG 86*, pages 264 – 267, 1986.
- [AMG95] G. Alonso, C. Mohan, and R. Günthör. Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In *[SKS95]*, pages 1 – 18, 1995.
- [Amm92] R.A. Ammar. Experimental-Analytical Approach to Derive Software Performance. *Information and Software Technology*, 34(4):229 – 238, April 1992.
- [And94] Rudolf Andersen. *A Configuration Management Approach for Supporting Cooperative Information System Development*. PhD thesis, The University of Trondheim, The Norwegian Institute of Technology, 1994.
- [AWS91] R.A. Ammar, J. Wang, and H.A. Sholl. Graphic Modelling Technique for Software Execution Time Estimation. *Information and Software Technology*, 33(2):151 –156, March 1991.

- [Bar89] Eric Barber. The Process Interaction Tool User Guide. Document IMSE R-5.1-3, STC plc, 1989. (V 5 : February 1991).
- [BC86] S. Berdal and Steinar Carlsen. PIP : Processes Interfaced through Ports. Master's thesis, IDT, NTH, Trondheim, Norway, 1986. DAISEE, Working Paper No 56.
- [BC89] E. Beslmüller and D.W. Conrath. The OSSAD Methodology. *ESPRIT '89 Conference Proceedings*, pages 865 – 877, December 1989.
- [Bei89] Heinz Beilner. Structured Modelling - Hetrogeneous Modelling. *Proc. European Simulation Multiconference*, 1989.
- [Bel87] T.E. Bell. Performance Engineering: Doing it "Later" on Large Projects. *CMG Transactions*, pages 75 – 81, winter 1987.
- [Bes88] Eduard Beslmüller. Office Modelling Based on Petri Nets. In *ESPRIT '88*, pages 977 – 987. North-Holland, 1988. Part 2.
- [BF87a] H Beilner and Stewing F.J. Concepts and Techniques of the Performance Modelling Tool, HIT. *Proceedings of the European Simulation Multiconference ESM'87*, 1987.
- [BF87b] T.E. Bell and A.M. Falk. Performance Engineering: Some Lessons From the Trenches. In *Proc. CMG 87*, pages 549 – 552, Dec. 1987.
- [BG92] Susanne Bødtker and Joan Greenbaum. Gender, Information Technology and the Design of Office Systems. In *Design of Information Systems: Things versus People*. Falmer Press, 1992. Editors: Green, E., Owen, J. and Pain, D.
- [BGHS91] Gordon Blair, John Gallagher, David Hutchison, and Doug Shepherd. *Object-Oriented Languages, Systems and Applications*. Pitman Publishing, 1991.
- [BH90] Eric Ole Barber and Peter H. Hughes. The Evolution of the Process Interaction Tool. In *Proceedings of the 17th Association of Simula Users Annual Conference*, Pilsen, Czechoslovakia, August 1990.
- [Bir79] G.M. Birtwistle. *Discrete Event Modelling on Simula*. Macmillan, 1979.
- [BMW88] H. Beilner, J. Maeter, and N. Weissenberg. Towards a Performance Modelling Environment: News on HIT. *Proceedings of the 4th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 1988.
- [BO91] Gunnar Brataas and Andreas Lothe Opdahl. Deriving Workload Models of Projected Software: A Case Study. IMSE Project report R6.6 – 9, SINTEF (University of Trondheim), July 1991. Version 1.
- [Boe81] Barry W. Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.

- [Boe88] Berry W. Boehm. A Spiral Model of Software Development and Enhancement. *Computer*, pages 61 – 72, May 1988.
- [BOVS92] Gunnar Brataas, Andreas Lothe Opdahl, Vidar Vetland, and Arne Sølvsberg. Information Systems: Final Evaluation of the IMSE. IMSE project deliverable D6.6 – 2, final version, The Norwegian Institute of Technology, The University of Trondheim, February 1992.
- [BOVS96] Gunnar Brataas, Andreas Lothe Opdahl, Vidar Vetland, and Arne Sølvsberg. A Method for Efficient Utilisation of Computing Resources. *In preparation*, 1996.
- [bpr95] What is BPR? listserve, May 1995.
- [Bra94] Gunnar Brataas. Performance Engineering of Task Processing Systems. ITOS Nettverk Trondheim, 2. juni 1994. Seminar.
- [Bru95] Lee Bruno. Administering Lotus Notes. *Open Computing*, 12(1):73 – 74, January 1995.
- [Bry93] Erik Brynjolfsson. The Productivity Paradox of Information Technology. *Communications of the ACM*, 36(12):66 – 77, December 1993.
- [BSH94] Gunnar Brataas, Arne Sølvsberg, and Peter Hughes. Integrated Management of Human and Computer Resources in Task Processing Organisations: A Conceptual View. In Jay F. Nunamaker and Ralph H. Sprague, editors, *27th Hawaii International Conference on Systems Science (HICSS-27). Minitrack: Modelling the Dynamics of Information Systems and Organisations*, volume IV, Information Systems: Collaboration Technology, Organizational Systems and Technology, pages 703 – 712, Maui, Hawaii, January 4 – 7 1994. IEEE Computer Society Press.
- [Buz86] Jeffrey P. Buzen. A Modeler's View of Workload Characterisation. In *Workload Characterisation of Computer Systems and Computer Networks*, pages 67 – 72. North-Holland, 1986. Editor: Guiseppe Serazzi.
- [Car95] Steinar Carlsen. Organizational Perspectives of Workflow Technology. Technical report, The Norwegian Institute of Technology, The University of Trondheim, May 1995.
- [CAS89] David W. Conrath, Valeria De Antonellis, and Carla Simone. A Comprehensive Approach to Modeling Office Organization and Support Technology. In B. Pernici et al., editor, *Office Information Systems: The Design Process*, pages 73 – 92. Elsevier Science Publishers B. V. (North Holland), 1989.
- [CH89] Thomas G. Cummings and Edgar F. Huse. *Organization Development and Change*. West Publication Company, fourth edition, 1989.

- [CH92] Kethy Center and Suzanne Henry. A New Paradigm for Business Processes. *IBM report*, pages 1 – 11, 1992.
- [Chu91] Lawrence Chung. Representation and Utilization of Non-Functional Requirements for Information System Design. In *[ABS91]*, pages 5 – 30, 1991.
- [Chu93] Lawrence Chung. *Representing and Using Non-Functional Requirements: A Process-Oriented Approach*. PhD thesis, University of Toronto, Toronto, Ontario, M5S 1A4, June 1993.
- [CKM⁺91] Lawrence Chung, Panagiotis Katalagarianos, Manolis Marakakis, Michalis Mertikas, John Mylopoulos, and Yannis Vassilou. From Information System Requirements to Design: A Mapping Framework. *Information Systems*, 16(4):429 – 461, 1991.
- [CMBN82] K.M. Chandy, J. Misra, R. Berry, and D. Neuse. The Use of Performance models in Systematic Design. In *Proc. National Computer Conference*, pages 251 – 256, Huston, June 1982.
- [Coa95] Workflow Management Coalition, May 1995. <http://www.aiai.ed.ac.uk/WfMC/>.
- [Cov90] Stephen R. Covey. *The 7 Habits of Highly Effective People; Powerful Lessons in Personal Change*. Simon & Schuster, 1990.
- [CS93] Maria Calzarossa and Giuseppe Serazzi. Workload Characterisation: A Survey. *Proceedings of the IEEE*, August 1993. Special issue on performance evaluation.
- [CVT⁺92] Eric Conquet, Alberto Valderruten, Rémi Tremoulet, Yves Raynaud, and Sandra Ayache. A Process Model of the Performance Evaluation Activity. In *Proceedings of the Summer Computer Simulation Conference*, Reno, USA, 27 – 30 July 1992.
- [Dat86] C.J. Date. *An Introduction to Database Systems*. The System Programming Series. Addison Wesley, fourth edition, 1986.
- [Dav93] Thomas Davenport. *Process Innovation: Reengineering Work through Information Technology*. Harvard Business School Press, 1993.
- [DBC88] A.M. Davis, E.H. Bersoff, and E.R. Comer. A Strategy for Comparing Alternative Software Development Life Cycle Models. *IEEE Transactions on Software Engineering*, 14(10):1453 – 1461, October 1988.
- [deM78] Tom deMarco. *Structured Analysis and System Specification*. Yourdon Press, 1978.
- [Den92] Peter J. Denning. Work Is a Closed-Loop Process. *American Scientist*, 80(4):314 – 317, July – August 1992.

- [Den94] Peter J. Denning. The Fifteenth Level. In *SIGMETRICS '94 Proceedings*, pages 1 – 4, Nashville, May 1994.
- [DM92] Elias Drakopoulos and Matt J. Merges. Performance Analysis of Client-Server Storage Systems. *IEEE Transactions on Computers*, 41(11):1442 – 1452, November 1992.
- [DoCS95] University of Twente Department of Computer Science. Workflow Management Project. WWW, November 1995. <http://wwwwis.cs.utwente.nl:8080/joosten/workflow.html>.
- [Dru91] Peter F. Drucker. The New Productivity Challenge. *Harvard Business Review*, pages 69 – 79, November – December 1991.
- [DS90] Thomas H. Davenport and James E. Short. The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review*, pages 11 – 27, Summer 1990.
- [Ecc91] Robert G. Eccles. The Performance Measurement Manifesto. *Harvard Business Review*, pages 131 – 137, January – February 1991.
- [EGR91] C.A. Ellis, S.J. Gibbs, and G.L. Rein. Groupware: Some Issues and Experiences. *Communications of the ACM*, 34(1):38 – 58, January 1991.
- [Fer78] Domenico Ferrari. *Computer Systems Performance Evaluation*. Prentice-Hall, Englewood-Cliffs, New Jersey 07632, 1978.
- [FHM⁺95] G. Franks, A. Hubbard, S. Majumdar, Petriu D., and C.M. Rolia, Woodside. A Toolset for Performance Engineering and Software Design of Client-Server Systems. *Performance Evaluation*, 24(1 – 2):117 – 136, November 1995.
- [Fin91] Anthony Finkelstein. A Neat Alphabet of Requirements Engineering Issues. In *ESEC '91, 3rd European Software Engineering Conference*, pages 489 – 491. Springer-Verlag, 1991.
- [Flø91] Kenneth Fløstrand. Estimating Organization Work: A Blood Bank Case Study. Master's thesis, The Norwegian Institute of Technology, The University of Trondheim, Dec 1991.
- [Fox89] G. Fox. Performance Engineering as a Part of the Development Life Cycle for Large-Scale Software Systems. In *Proc. 11th International Conference on Software Engineering*, pages 85 – 94, Pittsburgh, PA, May 1989.
- [FRCL91] Ellen Francik, Susan Ehrlich Rudman, Donna Cooper, and Stephen Levine. Putting Innovation to Work: Adoption Strategies for Multimedia Communication Systems. *Communications of the ACM*, 34(12):53 – 63, December 1991.

- [FRI95] FRISCO. Personal communication, March 1995. IFIP, WG 8.1, Task group: FRamework of Information System COnccepts.
- [FSZ83] Domenico Ferrari, Giuseppe Serazzi, and Alessandro Zeigner. *Measurement and Tuning of Computer Systems*. Prentice-Hall, Englewood Cliffs N.J. 07632, 1983.
- [Gal93] R.D. Galliers. Towards a Flexible Information Architecture: Integrating Business Strategies, Information Systems Strategies and Business Process Redesign. *Journal of Information Systems*, 3:199 – 213, 1993.
- [Ger93] Han Gerrits. Business Process Redesign and Information Systems Design: A Happy Couple? In *Information System Development Process*, pages 325 – 336. Elsevier Science Publisher B.V. (North-Holland), 1993.
- [GH91] Rebecca A. Grant and Christopher A. Higgins. Computerized Performance Monitors: Factors Affecting Acceptance. *IEEE Transactions on Engineering Management*, 38(4):306 – 315, November 1991.
- [GH95] Diimitrios Georgakopoulos and Mark Hornick. An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, (3):119 – 153, 1995.
- [Gil76] Arthur Gill. *Applied Algebra for the Computer Sciences*. Series in Automatic Computation. Prentice-Hall, 1976.
- [Gje93] Reidar Gjersvik. *The Construction of Information Systems in Organisations, An Action Research Project on Technology, Organisational Closure, Reflection and Change*. PhD thesis, The Norwegian Institute of Technology, 1993.
- [GKT93] Subashish Guha, William J. Kettinger, and James T.C. Teng. Business Process Reengineering: Building a Comprehensive Methodology. *Information Systems Management*, pages 13 – 22, Summer 1993.
- [GLW91] Jon Atle Gulla, Odd Ivar Lindland, and Geir Willumsen. PPP, An Integrated CASE Environment. In *[ABS91]*, pages 194 – 221, 1991.
- [Gru94] Jonathan Grudin. Groupware and Social Dynamics: Eight Challenges for Developers. *Communications of the ACM*, 37(1):92 – 105, January 1994.
- [GS77] Chris Gane and Trish Sarson. *Structured Systems Analysis: Tools and Techniques*. MCDONNELL DOUGLAS, 1977.
- [Gul93] Jon Atle Gulla. *Explanation Generation in Information Systems Engineering*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, September 1993.

- [Hal94] Kristin Halvorsen. Synliggjøring og Realisering av Forbedringsmuligheter i Virksomhetsprosesser. Master's thesis, The Norwegian Institute of Technology, The University of Trondheim, December 1994. In Norwegian.
- [Ham90] Michael Hammer. Reengineering Work: Don't Automate, Obliterate. *Harvard Business Review*, pages 104 – 112, July – August 1990.
- [Har88] David Harel. On Visual Formalisms. *Communications of the ACM*, 31(5):514 – 530, 1988.
- [Har91] H. J. Harrington. *Business Process Improvement*. McGraw-Hill, Inc., 1991.
- [HBP+88] Peter Hughes, Eric Ole Barber, Rob Pooley, Graham Titterington, and Chris Uppal. The Integrated Modelling Support Environment Design Study. Document SIMMER ICL229/1, STC plc, 1988.
- [HL84] P. Heidelberger and S.S. Lavenberg. Computer Performance Evaluation Methodology. *IEEE Transactions on Computers*, 33(12):1195 – 1220, Dec. 1984.
- [Hol91] Robert H. Holland. Integration of the Information Cycle. *Data Resource Management*, pages 6 – 18, Summer 1991.
- [Hor92] A. S. (editor) Hornby. *Oxford Advanced Learner's Dictionary: Encyclopedic Edition*. Oxford University, 1992.
- [Hug78] Peter Hughes. The Design and Use of Benchmarks for the Measurements of Computer System Performance. In *Proceedings of the Summer School on Computer Systems Performance Evaluation*, Sogesta, Urbino, Italy, June 1978.
- [Hug84] Peter Hughes. PILOT — A Synthetic Prototype Generator for Database Applications. In *Proceedings of the International Conference on Modelling Techniques and Tools for Performance Analysis*, Paris, May 1984. INRIA.
- [Hug88] Peter Hughes. SP principles. Technical report, STC Technology o59/ICL226/0, July 1988.
- [Hug89] Peter Hughes. IMSE Initial Design. Document IMSE D-1.2-1, STC plc, 1989. (V 1 : June 1989 : Deliverable D-1.2-1).
- [Hug93] Peter Hughes. A Modular Approach to System Structure and Performance Specification. Technical report, Modicum LTD, 1993. Draft.
- [Hug95] Peter Hughes. Hierarchical Performance Specification (single class). March 1995.
- [Hug96] Peter Hughes. Lecture Notes. Technical report, The Norwegian Institute of Technology, The University of Trondheim, 1996.

- [Hys91] William F. Hyslop. *Performance Prediction of Relational Database Management Systems*. PhD thesis, Computer Science Research Institute, University of Toronto, Toronto, Canada, M5S 1A1, September 1991. Technical Report CSRI-254.
- [Ide95a] Jon Iden. *Business Process Reengineering: Evaluating some Common Assumptions about the role of Information Technology*, volume 35 of *Reports in Information Science*. Department of Information Science, University of Bergen, Bergen, Norway, January 1995.
- [Ide95b] Jon Iden. *Workflow Management: What does the Litterature say*, volume 36 of *Reports in Information Science*. Department of Information Science, University of Bergen, Bergen, Norway, January 1995.
- [Iiv90a] J. Iivari. Hierarchical Spiral Model for Information System and Software Development. Part 1: Theoretical Background. *Information and Software Technology*, 32(6):386 – 399, July/August 1990.
- [Iiv90b] J. Iivari. Hierarchical Spiral Model for Information System and Software Development. Part 2: Design Process. *Information and Software Technology*, 32(7):450 – 458, September 1990.
- [Jab94] Stefan Jablonski. MOBILE: A Modular Workflow Model and Architecture. In *Proceedings of the Fourth International Working Conference on Dynamic Modelling and Information Systems*, Norordwijkerhout, The Netherlands, September 1994.
- [Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991.
- [JLP+95] J. Juopperi, A. Lehtola, O. Pihlajamaa, A. Sladek, and Veijalainen. Usability of Some Workflow Products in an Inter-Organizational Setting. In *[SKS95]*, pages 19 – 34, 1995.
- [JMSV92] Matthias Jarke, John Mylopoulos, J.W. Schmidt, and Y. Vassiliou. DAIDA: An Environment for Evolving Information Systems. *ACM Transactions of Information Systems*, 10(1):1 – 50, January 1992.
- [Joo95] Stef Joosten. Conceptual Theory for Workflow Management Support Systems. Technical report, Centre for Telematics and Information Technology, University of Twente, P.O. Box 217, 7500 AE Enschede, the Netherlands, July 1995.
- [Jør93] Håvard Dingstad Jørgensen. Performance Modelling of Task Processing Systems. Master's thesis, The Norwegian Institute of Technology, The University of Trondheim, December 1993.
- [Kan92] K Kant. *Introduction to Computer System Performance Evaluation*. McGraw-Hill, Inc., 1992.

- [KB95] Setrag Khoshafian and Marek Buckiewicz. *Introduction to Groupware, Workflow, and Workflow Computing*. Wiley, 1995.
- [Ker89] K. Kerzner. *Project Management*. Van Nostrand Reinhold, third edition, 1989.
- [KKP90] Steven E. Keller, Laurence G. Kahn, and Roger B. Panara. Specifying Software Quality Requirements with Metrics. In *[TD90]*, pages 145 – 163. 1990.
- [KLO+93] Rudolf K. Keller, Richard Lajoie, Marianne Ozkan, Fayez Saba, Xijin Shen, Tao Tao, and Gregor v. Bochmann. The Macrotec Toolset for CASE-based Business Modelling. *IEEE*, pages 114 – 118, 1993.
- [Kol86] Kenneth W. Kolence. An Overview of the Capacity Management Process. *IEEE*, pages 741 – 750, 1986.
- [Kot95] John P. Kotter. Why Transformation Efforts Fail. *Harvard Business Review*, pages 59 – 67, March – April 1995.
- [Kow95] Alexander Kowalski. Modelling of Workflow System Software for Performance Evaluation. Master's thesis, The Norwegian Institute of Technology, The University of Trondheim, June 1995.
- [Kro95] John Krogstie. *Conceptual Modeling for Computerized Information System Support in Organizations*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, August 1995.
- [LA94] Frank Leymann and Wolfgang Altenhuber. Managing Business Processes as an Information Resource. *IBM Systems Journal*, 33(2):326 – 348, 1994.
- [Lea65] H.J. Leavitt. Applied Organizational Change in Industry: Structural, Technological and Humanistic Approaches. In *Handbook of organizations*, pages 1144 – 1170. Rand McNally & Co., Chicago, 1965. Edited by James G. March.
- [LGNH94] Egil Lohndal, Tor Gunnar Gløppen, Morten Nygaard, and Carl Einar Halvorsen. Håndbok for Lisensarbeid i Undersøkelse og Produksjon. Intern rapport U&P-KP15 Lisensarbeid, Statoil, Januar 1994. In Norwegian.
- [Lin93] Odd Ivar Lindland. *A Prototyping Approach to Validation of Conceptual Models in Information Systems Engineering*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, June 1993.
- [Lot93a] Lotus. *Administrator's Guide*. Lotus Development Corporation, 55 Cambridge Parkway, Cambridge, MA 02142, 1993. Lotus Notes Release 3.

- [Lot93b] Lotus. *Getting Started with Application Development*. Lotus Development Corporation, 55 Cambridge Parkway, Cambridge, MA 02142, 1993. Lotus Notes Release 3.
- [LSKA95] Jens-Otto Larsen, Anne Helga Seltveit, Bo Kähler, and Jan Øyvind Aagedal, editors. *Proceedings of the 8th ERCIM Database Research Group Workflow on Database Issues and Infrastructure in Cooperative Information Systems*, Trondheim, NORWAY, August 1995. SINTEF and ERCIM.
- [LSS94] Odd Ivar Lindland, Guttorm Sindre, and Arne Sølvberg. Understanding Quality in Conceptual Modeling. *IEEE Software*, pages 42 – 49, March 1994.
- [LZGS84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance - Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Englewood Cliffs, New Jersey 07632, 1984.
- [MAD94] Daniel A. Menascé, Virgilio A. Almeida, and Larry W. Dowdy. *Capacity Planning and Performance Modelling*. Prentice Hall, 1994.
- [Mar92] Ronni T. Marshak. Requirements for Workflow Products. In D. Coleman, editor, *Groupware '92*, pages 281 – 285. Morgan Kaufman Publishers Inc., 1992.
- [Mar94a] Ronni T. Marshak. Perspectives on Workflow. In *[WF94]*, 1994.
- [Mar94b] Ronni T. Marshak. Workflow White Paper. In *Workgroup Computing Report*, pages 15 – 42. Patricia Seybold Group, 1994. Volume 16, Number X.
- [MBJK90] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: Representing Knowledge About Information Systems. *ACM Transactions of Information Systems*, 8(4):325 – 362, October 1990.
- [MC94] Thomas W. Malone and Kevin Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1):87 – 119, March 1994.
- [MCN92] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 18(6):483 – 497, June 1992.
- [Min90] Cydney Minkowitz. The Structure and Performance Specification Tool Design Document. Document IMSE R-4.1-1, STC Technology, 1990. (V 1 : June 1990 : Deliverable D-4.1-1).

- [MMWFF92] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The Action Workflow Approach to Workflow Management Technology. In *CSCW '92*, pages 1 – 10, Toronto, Canada, November 1992. ACM SIGCHI & SIGOIS.
- [Mor88] Gareth Morgan. *Organisasjonsbilder*. Universitetsforlaget, 1988. In Norwegian, translated from English.
- [MVH94] Cidney J. Minkowitz, Vidar Vetland, and Peter Hughes. A Modular Approach to System Structure and Performance Specification. In *The Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Vienna, Austria, 4 – 6 May 1994. Poster presentation.
- [Nix91] Brian Nixon. Implementation of Information System Design Specifications: A Performance Perspective. In Paris Kanellakis and W. Schmidt Schmidt, editors, *Database Programming Languages: Bulk Types & Persistent Data — Third International Workshop*, pages 149 – 168, Nafplion, Greece, August 27 – 30 1991. Morgan Kaufmann.
- [Nix93] Brian A. Nixon. Dealing with Performance Requirements During the Development of Information Systems. *Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 42 – 49, January 1993.
- [Nix94a] Brian A. Nixon. email communication, 24 October 1994.
- [Nix94b] Brian A. Nixon. Representing and Using Performance Requirements During the Development of Information Systems. In *Proceedings, Fourth International Conference on Extending Database Technology*, Cambridge, England, March 1994. Springer-Verlag. Published in the Lecture Notes in Computer Science series.
- [NN93] V.K. Narayanan and Raghu Nath. *Organization Theory: A Strategic Approach*. Irwin, Homewood, IL 60430, Boston, MA 02116, 1993.
- [Opd88] Andreas Lothe Opdahl. RAPIER - A Formal Definition of Diagrammatic Systems Specification. Master's thesis, IDT, NTH, Trondheim, Norway, 1988. DAISEE Working Paper No. 88.
- [Opd92] Andreas Lothe Opdahl. *Performance Engineering during Information System Development*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, November 1992.
- [OS92] Andreas Lothe Opdahl and Arne Sølvsberg. Conceptual Integration of Information System and Performance Modelling. In *Proceedings of IFIP WG 8.1 Working Conference on Information System Concepts – Improving The Understanding*, Alexandria/Egypt, 13–15 April 1992 1992.
- [OS94] Andreas Lothe. Opdahl and Guttorm Sindre. A Taxonomy for Real-World Modelling Concepts. *Information Systems*, 19(3), April 1994.

- [OSV93] Andreas Lothe Opdahl, Guttorm Sindre, and Vidar Vetland. Performance Considerations in Object-Oriented Reuse. In *Proceedings of the Second International Workshop on Software Reuse*, pages 142 – 151, Lucca, Italy, March 24 – 26 1993. IEEE Computer Society.
- [OVBS92] Andreas Lothe Opdahl, Vidar Vetland, Gunnar Brataas, and Arne Sølvsberg. CASE-Tool Support for Efficient Utilisation of Computing Resources. In *Proceedings of "The Third European Workshop on the Next Generation of CASE-Tools"*, Manchester, England, May 1992.
- [Par90] John Parkinson. Making CASE Work. In *In [SL90]*, chapter 15. John Wiley & Sons Ltd, 1990.
- [PBA⁺93] Ramon Puigjaner, Abdelmalek Benzekri, Sandra Ayache, Eric Conquet, Delphine Gazal, Omar Hjej, Alberto Valderruten, and Yves Raynauld. Estimation Process of Performance Constrains during the design of Real-Time & Embedded Systems. In Colette Rolland, François Bodart, and Corine Cauvet, editors, *Advanced Information Systems Engineering*, pages 629 – 648. Springer-Verlag, June 1993.
- [PBF⁺89] B. Pernici, F. Barbic, M Fugini, R. Maiocchi, J.R. Rames, and C. Rolland. C-TODOS: An Automatic Tool for Office System Conceptual Design. *ACM Transactions on Information Systems*, 7(4):378 – 419, October 1989.
- [PC86] David Lorge Parnas and Paul C. Clements. A Rational Design Process: How and Why to Fake it. *IEEE Transactions of Software Engineering*, SE-12(2):251 – 257, February 1986.
- [PR90] Barbara Pernici and Collette Rolland. *Automatic Tools for Designing Office Information Systems: The TODOS Approach*. Springer-Verlag, 1990.
- [PR95] Joe Peppard and Philip Rowland. *The Essence of Business Process Re-engineering*. The Essence of Management Series. Prentice-Hall, 1995.
- [PS85] James L. Peterson and Abraham Silberschatz. *Operating System Concepts*. Addison Wesley, second edition, 1985.
- [PVRS95] Ennio Pozzetti, Vidar Vetland, Jerome Rolia, and Giuseppe Serazzi. Characterizing the Resource Demands of TCP/IP. In *International Conference on High-Performance Computing and Networking 1995*, 1995.
- [PW93] Jeffrey Parsons and Yair Wand. Object-Oriented Systems Analysis: A Representation View. Working Paper 93-MIS-001, Faculty of Commerce and Business Administration, The University of British Columbia, Vancouver, BC, Canada, V6T 1Z2, January 1993.

- [Qua96] Quantum. Empire 1080 s, A High-Performance, Low-Power Disk Drive. <http://www.quantum.com/products/emp1080/>, February 1996.
- [Red94] Russell Redenbaugh. The New Common Sense. In [WF94], pages 13 – 24, 1994.
- [RS95] Jerome A. Rolia and Kenneth C. Sevcik. The Methods of Layers. *IEEE Transactions on Software Engineering*, 21(8):689 – 700, August 1995.
- [RV95] Jerome Rolia and Vidar Vetland. Parameter Estimation for Performance Models of Distributed Application Systems. In *Proceedings of CASCON '95*, Toronto, Canada, November 1995.
- [RVH95] Jerome Rolia, Vidar Vetland, and Greg Hills. Ensuring Responsiveness and Scalability for Distributed Applications. In *Proceedings of CASCON '95*, Toronto, Canada, November 1995.
- [RW94] Chris Ruemmler and John Wilkes. An Introduction to Disk Drive Modeling. *Computer*, pages 17 – 28, March 1994.
- [SB80] Connie Umland Smith and J.C. Browne. Aspects of Software Design Analysis: Concurrency and Blocking. *ACM Performance Evaluation Review*, 9(2), Summer 1980.
- [Sea69] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.
- [Sel94] Anne Helga Seltveit. *Complexity Reduction in Information Systems Modelling*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, December 1994.
- [Sil94] Bruce Silver. Automating the Business Environment. In [WF94], pages 129 – 154, 1994.
- [Sim94] Terje Simonsen. Hvordan redusere prosjektiden ved fokus på effektiv informasjonsflyt og minimum dokumentbehandling. In *Effektiv dokumentbehandling i offshore*, juni 1994. In Norwegian.
- [Sjø93] Wenche Sjøgren. Organizational Performance: A Blood Bank Case Study. Master's thesis, The Norwegian Institute of Technology, The University of Trondheim, December 1993.
- [SK93] Arne Sølvberg and David Chenho Kung. *Information Systems Engineering*. Springer-Verlag, 1993.
- [SKS95] Arne Sølvberg, John Krogstie, and Anne Helga Seltveit, editors. *Information Systems Development for Decentralized Organizations*, Trondheim, Norway, August 1995. IFIP, Chapman & Hall.

- [SL82] Connie Umland Smith and D.D. Loendorf. Performance Analysis of Software for an MIMD Computer. In *Proc. 1982 Conference on Measurements and Modelling of Computer Systems*, pages 151 – 162, 1982.
- [SL90] Kathy Spurr and Paul Layzell. *CASE on Trail*. John Wiley & Sons Ltd, 1990.
- [Smi86] Connie Umland Smith. Evaluation of Software Performance Engineering. In *Proc. FJCC*, pages 778 – 783, Dallas, Nov. 1986.
- [Smi90] Connie Umland Smith. *Performance Engineering of Software Systems*. Addison-Wesley, 1990.
- [Sø77a] Arne Sølvsberg. A Model for Specification of Phenomena, Properties, and Information Structures. Technical Report, IBM Research Laboratory, San Jose, California, 1977.
- [Sø77b] Arne Sølvsberg. On the Specification of Scenarios in Information System Design. Technical Report, IBM Research Laboratory, San Jose, California, 1977.
- [Sø79] Arne Sølvsberg. Software Requirement Definition and Data Models. In *Proceedings of the 5th International Conference on VLDB*, pages 111–118, 1979.
- [Sø80] Arne Sølvsberg. A Contribution to the Definition of Concepts for Expressing Users' Information Systems Requirements. In P. P. Chen, editor, *Entity-Relationship Approach to Systems Analysis and Design*. North-Holland, 1980.
- [ST89] Scott D. Sink and Thomas C. Tuttle. *Planning and Measuring in Your Organization of the Future*. Industrial Engineering and Management Press, 1989.
- [Sta92] Statoil. Konsept for saksbehandling. Technical report, Statoil, 26. August 1992. In Norwegian.
- [Sta93] Statoil. Replikering hos Statoil. Intern rapport, Statoil, December 1993. In Norwegian, partially in English.
- [Sta94a] Statoil. *Telefonkatalog for Statoil*. Statoil, Juli 1994. In Norwegian.
- [Sta94b] Statoil. Telextrafikk i GASS M. Intern rapport, dokumentasjon fase 1, Statoil, April 1994. In Norwegian.
- [Sta94c] Statoil. Telextrafikk i GASS M. Intern rapport, dokumentasjon fase 2, Statoil, Mai 1994. In Norwegian.
- [Str94] Straussman. Information Technology and Organizational Effectiveness. Tutorial on Hawaii International Conference on Systems Sciences, Maui, Hawaii, January 1994.

- [SW93] Connie Umland Smith and Lloyd G. Williams. Software Performance Engineering: A Case Study Including Performance Comparison with Design ALternatives. *IEEE Transactions on Software Engineering*, 19(7):720 – 741, July 1993.
- [SW94] Connie Umland Smith and Bernie Wong. SPE Evaluation of a Client/Server Application. In *CMG94 Proceedings*, pages 528 – 540, December 4 –9 1994.
- [TD90] Richard H. Thayer and Merlin Dorfmann. *Tutorial: System and Software Requirements Engineering*. IEEE Computer Society Press, 1990.
- [TK84] Howard M. Taylor and Samuel Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 1984.
- [Tri88] L. L. Tripp. A Survey of Graphical Notations for Program Design — An Update. *ACM SIGSOFT Software Engineering Notes*, 13(4):39–44, October 1988.
- [TW95] Thomas Tesch and Jürgen Wäsch. Research Issues in Workflow Systems. In *[LSKA95]*, 1995.
- [Twi90a] Twinco. Functional System Design Report. Project PAS/SQL Blood Bank Version 1.1, Twinco, December 1990.
- [Twi90b] Twinco. Requirements Specification. Project PAS/SQL Blood Bank Version 1.0, Twinco, March 1990. In Norwegian.
- [Twi91] Twinco. Computer System Design Report. Project PAS/SQL Blood Bank Version 1.0, Twinco, February 1991. In Norwegian.
- [Vet93] Vidar Vetland. *Measurement-based Composite Computational Work Modelling of Software*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, August 1993.
- [VHS93a] Vidar Vetland, Peter Hughes, and Arne Sølvberg. A Composite Modelling Approach to Software Performance Measurement. In *Proceedings of SIGMETRICS '93*, pages 275 – 276, Santa Barbara, USA, May 10 – 14 1993. Extended abstract.
- [VHS93b] Vidar Vetland, Peter Hughes, and Arne Sølvberg. Improved Parameter Capture for Simulation Based on Composite Work Models of Software. In *Proceedings of the 1993 Summer Computer Simulation Conference*, Boston, July 19 – 21 1993.
- [VLP95] Jari Veijalainen, Aarno Lehtola, and Olli Pihlajamaa. Research Issues in Workflow Systems. In *[LSKA95]*, 1995.
- [Von90] Roland Vonk. *Prototyping, The effective use of CASE Technology*. Prentice Hall, 1990.
- [Wal93] Sten Waløen. Bedre telex til GASS M. Arbeidsversjon 8, Avenir, oktober 1993. In Norwegian.

- [Wei96] Reinhold Weicker. Dhrystone Benchmark Results. Available via anonymous ftp from "ftp.nosc.mil" in directory "pub/aburto", file "dhry.tbl", January 1996.
- [WF94] Thomas E. White and Layna Fischer. *The Workflow Paradigm*. Future Strategies Inc., Book Division, Alameda, California, 1994.
- [Wil93] Geir Willumsen. *Executable Conceptual Models in Information Systems Engineering*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, November 1993.
- [Xen91] Nick Xenios. The Structure and Performance Specification Tool User Guide. Document IMSE R-4.2-1, BNR Europa Ltd, 1991. (V 1 : October 1991 : Deliverable D-4.1-2).
- [Yan89] Jianhua Yang. *Computer-Based Document Processing in Office Information Systems*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, October 1989.
- [Yan93] Mingwei Yang. *COMIS — A Conceptual Model for Information Systems*. PhD thesis, The Norwegian Institute of Technology, The University of Trondheim, August 1993.
- [Yav94] David Yavin. Optimizing Notes Replication. *BYTE*, pages 201 – 202, September 1994.
- [Yu94] Eric Yu. *Modelling Strategic Relationships For Process Reengineering*. PhD thesis, University of Toronto, December 1994.
- [Zav84] Pamela Zave. The Operational Versus the Conventional Approach to Software Development. *Communications of the ACM*, 27(2):104 – 118, February 1984.

Index

- Abstract virtual machine, 37
- Actor, 10, 130
- Batch load, 32
- Blood Bank Case Study, 91
 - Branching probabilities, 98
 - Database platform, 101
 - Introduction, 5
 - Performance requirements, 97
 - Projected application, 97
 - Resource demands, 100
 - Summary of findings, 106
 - System boundaries, 93
 - System platform, 100
 - Transfusion process, 6
 - Work model, 103
 - Workload, 94
- Bounds analysis, 21
- BPR, 114
 - Principles, 116
- Compactness, 39
- COMPLEMENT, 62
- Complexity specification, 30
- Computerised information systems, 11
- Contention modelling, 19
- Data model
 - In PPP: PhM, 204
 - In SP, 39
- Distributed models, 44
- Dynamic performance model, 19
 - Bounds analysis, 21
 - Petri nets, 22
 - Queueing networks, 19
- Effectiveness, 13
- Efficiency, 13
- Existing system, 11
- FESC, 21
- Further work, 195
- Gas Sales Telex Adm. Case Study, 139
 - Computerised solution
 - Resource demands, 183
 - Contract specialist, 163
 - Resource demand, 164
 - SP model, 163
 - Introduction, 139
 - Lotus Notes
 - Resource demands, 183
 - Lotus Notes client-server, 173
 - Customisation, 177
 - SP model, 173
 - Overall manual process model, 142
 - Replication architecture, 199
 - SP model, 176
 - Sales telex administration org., 144
 - Performance requirements, 146
 - Summary of findings, 165
 - Workload characterisation, 145
 - Similarities between manual and computerised solution, 188
 - Statoil organisation, 144
 - System boundaries, 141
 - Telex secretary, 147
 - Process model, 147
 - Resource demands, 156
 - SP model, 149
 - Work model, 154
 - Workflow system boundaries, 170
 - Workflow system workload, 172
- Group, 129
- HIT, 63
- Incremental development, 50
- Information systems, 11
 - Computerised, 11
 - Hierarchies, 122
 - Office, 113
- Layered Queueing Networks, LQMs, 60

- Load, 32
 - Batch, 32
 - Terminal, 32
 - Transaction, 32
- Local realities, 111
- Lotus Notes, 126
 - Field, 127
 - Replication, 199
 - View, 127
- LQMs, Layered Queueing Networks, 60
- Major contributions, 193
- Method for Performance Engineering, 69
 - Boundaries of discussion, 69
- Module specification, 37
- Non-distributed models, 44
- Non-functional requirements, 47
- Office information systems, 113
 - OSSAD, 113
- Operation, 11, 29, 37
 - Typing, 41
- Operational development, 51
- Organisation, 129
- Organisational images, 110
- OSSAD, 113
- Performance, 13
 - Measures, 16, 21
 - Specification, 35
- Performance engineering, 47
 - Motivation, 54
 - Qualitative, 67
 - Quantitative, 47
- Performance model, 15
 - Closed, 18
 - Cycle, 23
 - Different parts, 29
 - Dynamic, 19
 - Open, 18
 - Static, 25
 - Validation, 23
 - Verification, 23
- Petri nets, 22
- PhM, 204
- Platform, 79
- PLD, 209
- PPP, 203
 - Case tool, 210
 - Languages, 203
 - PhM, 204
 - PLD, 209
 - PrM, 205
 - Method, 53
- PrM, 205
- Projected system, 11
- Prototyping, 50
- Queueing Networks, 19, 22
- Replication arch. for Lotus Notes, 199
- Resource, 10, 130
 - Differences, human, comp. res., 135
 - Similarities, human, comp. res., 135
- SP, 25
 - Compactness, 39
 - Data model, 39
 - Distributed models, 44
 - Module specification, 37
 - Non-distributed models, 44
 - Opdahl's extensions, 63
 - Operation, 29
 - Unit of work, 30
 - Vetland's extensions, 64
 - Work, 30
 - Workspace, 40
- SPE, 57
- Spiral Model, 53
- Static performance model, 25
 - Interaction with CASE-tools, 63
 - Tradeoffs, 64
- Subsystem, 35
- Summary of limitations, 195
- System, 35
 - Closed, 18
 - Existing, 11
 - Open, 18
 - Projected, 11
- Terminal load, 32
- Transaction load, 32
- Transformational development, 51
- Unit of work, 30
- Waterfall model, 48

- Work, 30
 - Complexity specification, 30
 - Problems with characterisation, 31
- Work, unit vector, 30
- Workflow, 118
 - Ad-hoc, 133
 - IPO paradigm, 123
 - Production, 133
 - Speech act based paradigm, 123
 - Template, 133
 - Typical roles, 120
 - Workflow modelling, 123
- Workflow Reference Model, 125
- Workflow system, 118
 - Workflow Reference Model, 125
 - Two important subsystems, 132
- Workload, 32
- Workspace, 40
- World, 82