Design of adaptable simulation models.

POHL, Thomas.

REFERENCE

Design of Adaptable Simulation Models

Thomas Pohl

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University
for the degree of Doctor of Philosophy

September 2006

Sheffield Hallam University

Materials & Engineering Research Institute

The undersigned hereby certify that they have read and recommend to the Faculty of Arts, Computing, Engineering and Sciences for acceptance a thesis entitled **"Design of Adaptable Simulation Models"** by **Thomas Pohl** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy.**

Dated: September 2006

Research Supervisors: _____

Prof. Terrence Perera

_____

Dr David Clegg

Examining Committee: _____

Dr Tillal Eldabi

_____

Prof. Sameh Saad

Sheffield Hallam University

Date: **September 2006**

Author: **Thomas Pohl**

Title: **Design of Adaptable Simulation Models**

Department: **Materials & Engineering Research Institute**

Degree: **Ph.D.** Convocation: **October** Year: **2006**

Permission is herewith granted to Sheffield Hallam University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

THE AUTHOR ATTESTS THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED.

Signature of Author

Contents

List of Tables

List of Figures

Abstract

In today's world, with ever increasing competition, modelling and simulation proves to be a very helpful tool. Many methodologies exist to help build a simulation model from scratch. In terms of adaptability, most current attempts focus on either the operational side, ie the automated integration of data into a model, or the creation of new software. However, very few attempts are being made to improve the adaptability of shelved models built in existing simulation software. As a result, there is a certain reluctance, in some areas, to use simulation to its full potential.

Based on these facts, it is obvious that anything, which makes reuse of simulation models easier, can help improve the use and spread of simulation as a valuable tool to maintain a company's competitiveness. In order to find such a solution, the following issues are looked at in this thesis: The changes to a simulation model that constitute the biggest problem, ways to minimise those changes, and possibilities to simplify the implementation of those changes.

Those factors are evaluated, first by investigating current practices of building adaptable simulation models via a literature review, then the most difficult changes to implement in a simulation model, and the most frequent types of simulation software, are identified by means of interviews and questionnaire surveys. Next, parameters describing the adaptability of a simulation model are defined. In a further step, two of the most widely used simulation packages are benchmarked against a variety of tasks, reflecting the changes most frequent to models. The benchmarking study also serves to define and test certain elements regarding their suitability for adaptable models. Based on all those steps, model building guidelines for the creation of adaptable simulation models are developed and then validated by means of interviews and a framed field experiment.

The interviews and questionnaire reveal that deleting is the easiest task and modifying the most complicated, while handling devices are the most difficult element to modify. The results also show that simulators (eg Arena) are the most widespread type of simulation software. The benchmarking showed that Arena is overall more adaptable than Simul8, and confirms the findings from the user survey. Also, it shows that sequencing is very helpful for modifying models, while the use of sub-models decrease the adaptability. Finally, the validation proves that the model building guidelines substantially increase the adaptability of models.

Preface

This thesis is submitted to the Faculty of Arts, Computing, Engineering and Sciences of Sheffield Hallam University for the degree of Doctor of Philosophy

I would like to express my deepest gratitude and appreciation to my supervisors, Dr. Terrence Perera and Dr. David Clegg, for their guidance and constructive criticism during the course of this study. I would like to thank my family for their support. I would also like to thank Sue Raynor and all the administrative staff within the School of Engineering/ the Faculty of Arts, Computing, Engineering and Sciences for their help and support. My appreciation also extends to all those individuals who participated as interviewees in this study, especially Ruby Lau, Anna Lassila, Lionel Joseph and Claire Westby from Sheffield Hallam university, Edwin Valentin from Systems Navigator and the postgraduate students at the TU Delft in the Netherlands.

The results obtained during the course of this research are to the best of my knowledge original, except where reference is made to the work of others.

T. Pohl

Sheffield, October 31, 2006

Chapter 1

General Introduction

1.1 Background to research

With the increasing competition on the global market, companies are forced to gain an advantage over their competitors in order to stay successful. Customers, on the other hand, want to get the best possible product for the lowest price, thereby fuelling the competition among suppliers in their quest to produce the best possible product or service at the lowest price over all. In order to minimize their costs and thus gain and/or keep their competitive advantage in applying economies of scale, or finding a niche market, or offering higher quality products/services, the companies turn to modelling. This means that modelling is used as a tool to improve current processes or products/services. Simultaneously, simulation thus serves as a tool improve or maintain one's competitive edge.

Modelling is a simple term that covers a huge domain. Architects use small-scale models of their creations to give their customers a clearer picture of what the finished building will look like. Aeronautics companies and car manufacturers use small-scale models to test their creations' behaviour and influence on air currents. Physicists, astronomers and mathematicians use mathematical models to describe their theories and findings. In fashion, human models are used to show the designers' new creations to the interested public.

Another aspect of modelling is simulation, where processes from the real world are reconstructed on a computer in order to experiment without having to actually use the real world system. Simulation can be used in various forms, among which are: Simulation of mechanical stresses of a material or component, and the simulation of processes. The former is used in the design or redesign of parts or components, where testing the actual

part would be too costly or impossible, such as in the design of a new aircraft wing. The second type of simulation is used in the design or modification of a process, be it in manufacturing or services, where the process does not exist yet, or where experimenting with the actual system would be too costly or difficult, e.g. when it would interfere with ongoing production.

While some simulation projects are of relatively short duration - days or weeks - and fairly simple, other simulations may have a much larger time frame. In the aeronautical industry, for example, where simulation is extensively used, a project, such as a new airplane, has a time frame of up to 30 or 40 years from the first concept. Such a project may start as a concept or study, then move on to the production of prototypes. Next, serial production starts and modifications to the plane may be introduced every few years as new technology becomes available. After production of the airplane as such stops, production of spare parts may still go on, as the existing aircraft have to be serviced until the airplane is phased out. Not only does such a project span over long time periods and consist of very diverse stages, such projects are also very complex due to their sheer size, the number of components, people and organisations involved.

Due to large projects undergoing different phases, any simulations used over the course of the project life cycle need to be adapted to those changes. In manufacturing for example, as new technology or a new machine is introduced to the plant, this change needs to be reflected in the simulation and the same goes for a change of goods produced.

1.2 Justification for research

Currently simulation models are mostly used on a one-off basis. As McLean and Leong (2002) pointed out, *"simulations are often developed from scratch"*. This means that models are constructed for a certain purpose, such as the design of new systems or the improvement of an existing one, and then discarded. Also, building a simulation model is a lengthy process. This is due to the need to analyse the existing system and/or to gather all the necessary data to build a valid, credible model.

On the other hand, further use for the model may arise later. Chance et al. (1996) pointed out that using such single-use models on an on-going basis leads to a lot more additional work. Yet, the reuse of existing models stems from the idea of spreading the cost of building the model through prolonging its use. However, due to the changes

occurring to the system over the course of the project as mentioned above, it becomes necessary for the simulation models to be adaptable. This means that there is a need to modify an existing simulation to fit a changed system and, most of all, a need to modify it so that it delivers usable answers to a new problem.

Another dimension of change within a model, even one for single-use, is that, as the understanding of the system under study improves and more data becomes available, this might also lead to a change of focus of the simulation study. The building of a simulation model, ie of its logic, generally starts before all necessary data have been gathered. As a result, the main goals of the simulation study are rather crude and get refined over time, as more and more data becomes available. However, this increase in knowledge and understanding of the system under study can make the original simulation objective obsolete, or reveal that it needs to be somewhat modified to gain or retain validity. As a result, changes within a single simulation model might need to be implemented by the analyst.

The need for adaptable simulation models is well recognized in literature.
McLean and Leong (2002) proposed a modularization of simulation models to help in that respect. Pidd (2002) suggested various reuse strategies. Reese and Wyatt (1987) defined software reuse as *"the isolation, selection, maintenance, and use of software components in the development and maintenance of a software project"*. de Ruiter et al. (2000) suggested a concept for an integrated simulator for recurring decisions, and
Herrmann et al. (2000) reviewed concepts of adaptability and put forth suggestions to measure this adaptability.

Hlupic (2000) conducted a survey on academic and industrial users of simulation software in which she showed that more than half of academic and a quarter of industrial users had problems in modelling associated with a lack of adaptability and software limitations of the package used.

However, very little research has been undertaken so far to address the adaptability of simulation models in terms of allowing future reuse or modification of existing models.

1.3 Focus of research

Although this need for adaptability has been recognised, the impact of a necessary change on the overall project has not been examined yet, nor what types of change constitute the biggest problem to the simulation expert.

From this, the following questions arose:

i **What change or changes to a simulation project constitute the biggest problem, at its implementation, for simulation software users?**
ii **How can those changes be minimized?**
iii **How can the implementation of changes to a simulation project be simplified?**

Together, these questions form the basis for the research work proposing a novel design of adaptable simulation models in manufacturing, based on a widely used simulation package.

To this end, it was necessary to:

i Investigate current practices of building adaptable simulation models in industry
ii Identify the most difficult and/or most frequent changes to implement in a simulation study, irrespective of the software used,
iii Identify parameters which describe the adaptability of simulation models
iv Conduct a benchmarking study of some of the leading simulation software packages towards their inbuilt adaptability
v Develop model building guidelines for the creation of adaptable simulation models
vi Develop a prototype adaptable simulation model and validate its design.

1.4 Outline of thesis

This thesis is divided into 8 parts, consisting of a general introduction, a literature review, research methodology, development and conduct of the research, analysis of the results and a conclusion.

1.4.1 Chapter 1 – Introduction This chapter contains an introduction to the subject of research and an overview of each of the chapters of this thesis.

1.4.2 Chapter 2 – Literature Review In the literature review, an overview of the subject area is given and the strengths and weaknesses of the referenced material are discussed. The questions arising from this analysis form the main research questions.

1.4.3 Chapter 3 – Research Methodology Chapter 3 reviews the methodologies used for the research. These include interviews, surveys, benchmark studies, action research and case studies.

1.4.4 Chapter 4 – Interviews and Questionnaire Survey This chapter gives an overview of the data generated by means of interviews and questionnaire surveys and how the research methodologies were put into practice to obtain that data. It also shows the data via the survey and the analysis thereof.

1.4.5 Chapter 5 – Benchmarking The chapter on benchmarking investigates the overall adaptability of two simulation packages and compares them to one another. Also, the extent to which the use of sub-models helps understanding a model is tested through benchmarking.

1.4.6 Chapter 6 – Modified Model Building Guidelines This chapter introduces a methodology specifically aimed at adaptable simulation models and their construction. This methodology is based on the results obtained from the preceding chapters.

1.4.7 Chapter 7 – Validation of the Guidelines In this chapter, the validation of the modified model building guidelines is looked at. For the validation, interviews and experiments are used.

1.4.8 Chapter 8 – Findings and Conclusion In Chapter 8, the results obtained are analysed and put into the context of the overall aim.

Chapter 2

Literature Review

This chapter investigates various theories on and approaches to model building, different types of adaptability generally encountered in simulation, along with their definitions. It also looks at the adaptability of simulation tools, how such adaptability can be measured and the status quo on surveys of simulation software.

2.1 Introduction

In this chapter, the current practices of building adaptable simulation models and, more generally, ways to render simulation models more adaptable and reusable are investigated. In a first part, an overview of general model building principles and practices as described in literature is given. This is important to help place changes to the modelling process in the right context. The second part focuses on the different approaches to aiding the simulation expert in making simulation models more adaptable and reusable.

2.2 Definitions

The New Shorter Oxford English Dictionary gives the following definitions:

> Model: "Representation of structure, an abstract; A three-dimensional representation of an existing person or thing or projected structure, showing the proportions and arrangement of its component parts; A simplified description of a system, process, etc., put forward as a basis for theoretical or empirical understanding; a conceptual model or mental representation of something." (The New Shorter Oxford English Dictionary).

- Simulation: "1) The action or an act of simulating something; computing a model produced by this means" (The New Shorter Oxford English Dictionary).

In the context of this research, the term model refers to a simplified, conceptual representation of a system or process used to gain a better understanding. And, as the definitions quoted above indicate, modelling in this case is the creation of a computerized representation of a simplified process or system for gaining theoretical and/or empirical understanding.

Banks and Carson (1984) defined simulation as *"the imitation of the operation of a real-world process or system over time. ... Simulation involves the generation of an artificial history of a system, and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system."*

With their definition, Banks and Carson introduced another factor: time. Not only is time inherent to simulations, thus signifying constant change in the output of the model, which then leads to a necessity of running a model various times to achieve statistic meaning. But the notion of time and change, especially of the real system, may also give rise to the necessity of implementing a change in the simulation model as a whole.

2.3 General principles

Over the course of years, simulation experts have proposed different approaches to building simulation models, featuring a varying number of phases. In this sub-chapter, the main approaches to simulation modelling are presented to give the reader a clearer picture of the general model building principles and practices. Also, the approaches described below serve as the basis for a adaptability-orientated approach.

2.3.1 Law and Kelton Law and Kelton (2000) proposed a ten step approach to model building (see Fig. 2.1 on page 9).

1 Problem formulation and planning of the study. This step includes discussion of the objectives, the specific questions to be answered by it, the components to be modelled, the time frame, performance measures, the software to be used and the overall scope of the model.

2 Data collection and model definition. At this stage, information about the system and data from it are collected and a conceptual model built. This conceptual model

should include the goals of the study, the performance measures, detailed descriptions of the system and its subsystems in bulleted form, what was simplified and why, summaries of data and sources of information.

3 Validity of the conceptual model? This is an important step to assure the model's assumptions are complete and correct.

4 Programming and program verification. Once the conceptual model's validity is established, the model is translated into computer code, either through a programming language (e.g. Fortran or C), or through simulation software (e.g. Arena™ or AutoMod™) and debugged.

5 Pilot runs. At this phase, pilot runs are made to establish its validity in the next step.

6 Validity of programmed model? Here, the pilot runs are used to compare the model to the system if possible. More generally, the program should be reviewed for correctness and to determine the model factors which have most influence on the performance measures.

7 Design of experiments. Here, the appropriate number of independent simulation runs, length of runs and warm-up periods for each configuration of interest are established.

8 Production runs. At this step, the experiments defined in the previous phase are run

9 Output data analysis. The data obtained from the production runs is analysed to compare alternative systems or systems configurations and their overall performance.

10 Documentation, presentation and use of results. This includes documenting the assumptions, the program and the results obtained from the study. The presentation may be made using animation and should also contain discussion of the model building and validation process.

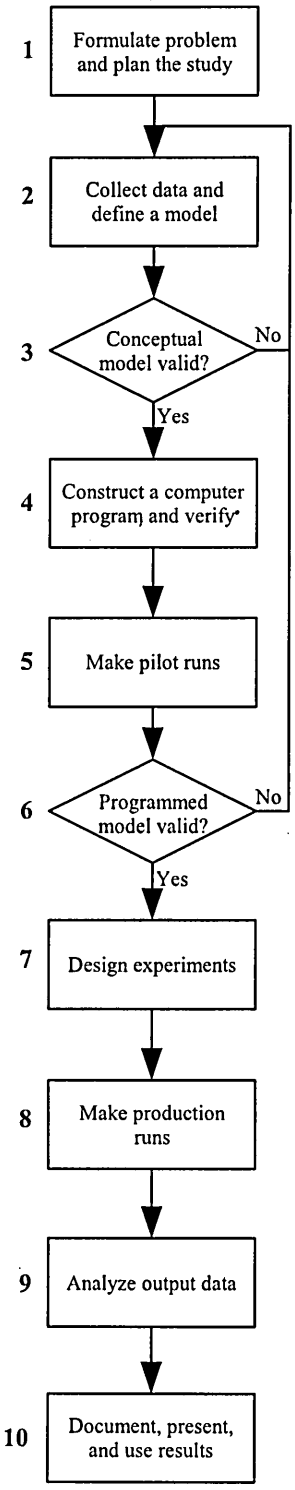

Figure 2.1: Steps in a simulation study (Law and Kelton 2000)

2.3.2 Balci Balci (1990) defined the simulation life cycle as consisting of 10 phases, and as a reiterative process. In order to attempt an answer to this question, the life cycle of a simulation study needs to be analysed step by step, looking at the consequences of a change at any stage for the other processes in the life cycle. This analysis is based on the life cycle of a simulation study by Balci (1990), depicted in Fig. 2.2 on page 12.

1 The first process, ***problem formulation***, links a communicated problem to a formulated problem and is the first process involving an analyst. This is the stage at which the problem gets defined to the point where "specific research action" can be taken (Balci 1990).

2 The ***investigation of solution techniques*** constitutes the second process. Depending on the outcome from the first phase, simulation or another analysis tool is chosen.

3 The ***system investigation*** is the next process and deals with defining the objectives of the simulation model and serves as the basis for building a conceptual model.

4 The next phase in the life cycle is ***model formulation***, where conceptual models of the system under study are built, based on the objectives outlined in the system investigation. At this stage the decision about the underlying data is also made, i.e. whether to use a probabilistic or deterministic model. This phase also consists of building the input data, for example through sampling of actual data to extrapolate the distribution to then feed the data into the model.

5 During the ***model representation*** phase, the conceptual model created is adapted for different audiences, such as experts, nontechnical people etc, who all need different representations, e.g. pseudo code, flowcharts or structured graphs.

6 ***Programming***. Once the model representation has been achieved, the programmer turns his representation into computer code and debugs it, thus constructing the simulation model to be run. With modern simulators, e.g. Arena™, the programming stage has been greatly simplified, as graphical modules are used, and no line by line programming is, over all, necessary. This makes both the programming and the debugging much faster.

7 Once the simulation model is fully programmed, and therefore executable, the next stage is the ***design of experiments***, i.e. to decide on the best combination of parameters to achieve the desired results.

8 The experiments to be run having been laid out, the actual ***experimentation*** can take place.

9 At the ***redefinition*** stage, the existing model is changed, to adapt it to the current situation, and to provide for new experiments. Balci differentiates between five different reasons for possible changes to the model: (1) updating the model to the current, i.e. new, stage of the actual system, (2) altering it to obtain another set of results (3) change it for maintenance, (4) adapt it for another use and (5) redefining a new system for studying alternatives.

10 ***Presentation of simulation results***. Parallel to redefining the model, the simulation results are interpreted and presented

However, apart from these sequential steps, a life cycle of a simulation model consists of reiterative processes. Such reiteration is based on the criteria of credibility assessment, i.e. the verification, validation and testing (VV&T) of the model. Such assessments are represented by two-way arrows in Fig. 2.2 on the following page, while the simple sequential stages are represented by dotted arrows. Overall, these two-way arrows show that if a discrepancy at any phase is discovered, this can trigger changes at phases of greater or lesser detail.

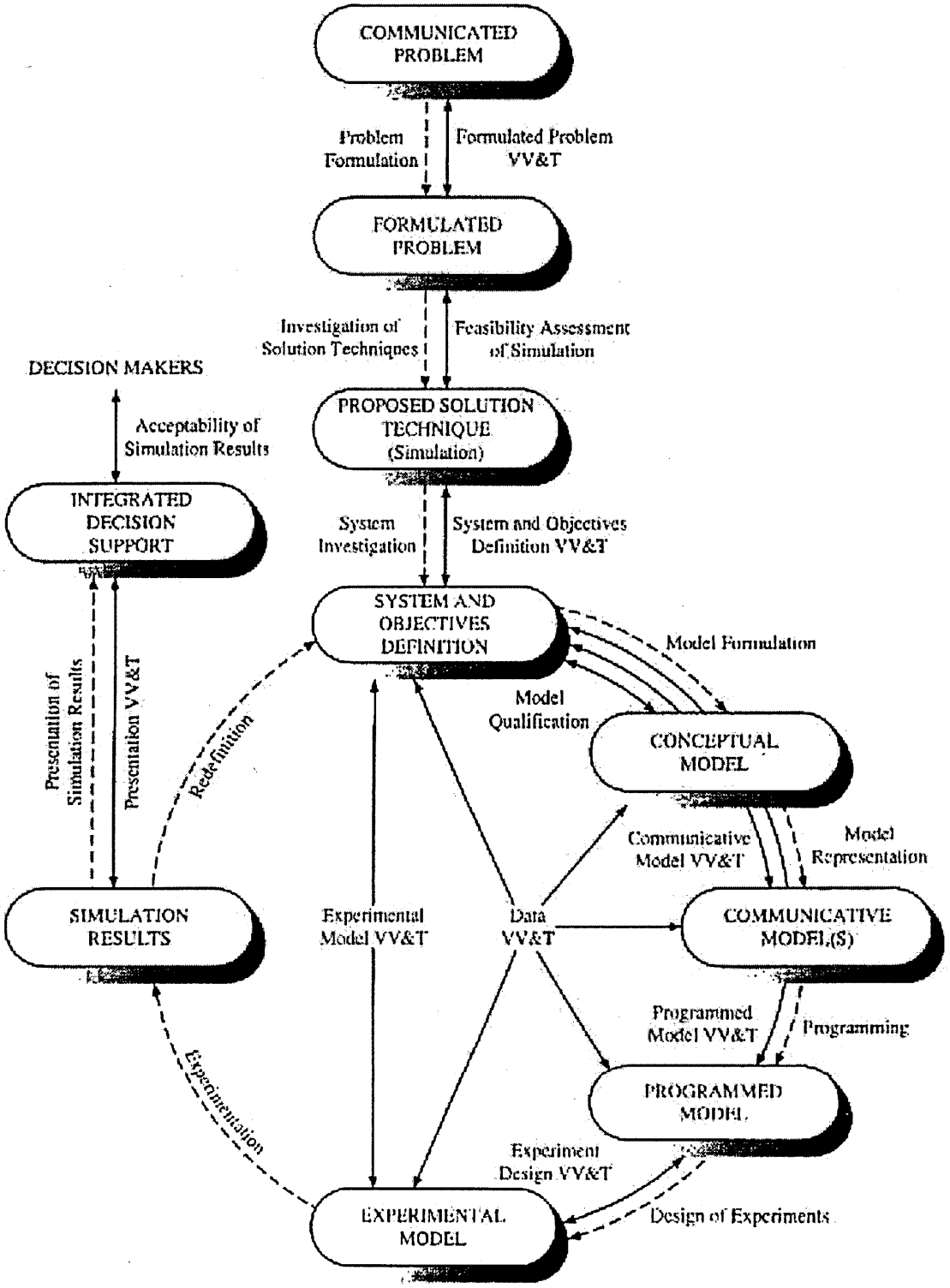

Figure 2.2: The life cycle of a simulation study (Balci 1990)

2.3.3 Banks and Carson Banks and Carson (1984) showed a slightly different simulation modelling process. In this 12-step flowchart, (see Fig. 2.3 on the next page), the verification and validation stages are firmly engrained in the process.

1 ***Problem Formulation***. At this stage, a problem with an existing system or one under design is stated. On some occasions, there might be a need to reformulate the problem at a later stage.

2 ***Setting of objectives and overall project plan***. Here, the modeller establishes which tool should be used (e.g. simulation or another analytical tool) as well as a set of assumptions. Also, the goals and definition of the project are defined, along with methods to evaluate the effectiveness of alternatives.

3 ***Model Building***. This refers to devising a conceptual model of the system under study. This means a more or less simplified, abstracted representation of the processes to be studied consisting of the essential components only.

4 ***Data Collection***. Data about the system is gathered if possible. If no data is available, rules of thumb or personal experience are used to approximate distributions. Data requirements may change as the model and its complexity progresses.

5 ***Coding***. Once the conceptual model is established and the data gathered, the conceptual model is translated into executable code, i.e. into the actual simulation program.

6 ***Verification***. This stage is necessary to assure the coded model does not contain any bugs and performs as expected and intended. If this verification fails, the code needs to be reviewed and corrected until it passes verification.

7 ***Validation***. Here, the focus is on verifying whether the model represents the system to be modelled with sufficient accuracy. If the model does not represent the behaviour of the real system with sufficient accuracy, the conceptual model and/or the data gathered needs to be reviewed.

8 ***Experimental design***. This involves designed the alternatives to be studied, along with the number of replications, the replication length and warm-up periods.

9 ***Production runs and analysis***. This phase consists of running the experiments designed in the previous step and analysing the results thus obtained.

10 ***More runs?*** At this stage, the modeller has to verify whether the results obtained thus far are sufficient, or whether more runs or different experiments have to be run.

11 ***Document study and report results***. The results obtained from the simulation runs, once analysed, have to be documented and reported/presented to the problem owner so that the results can be implemented in the actual system. This is also

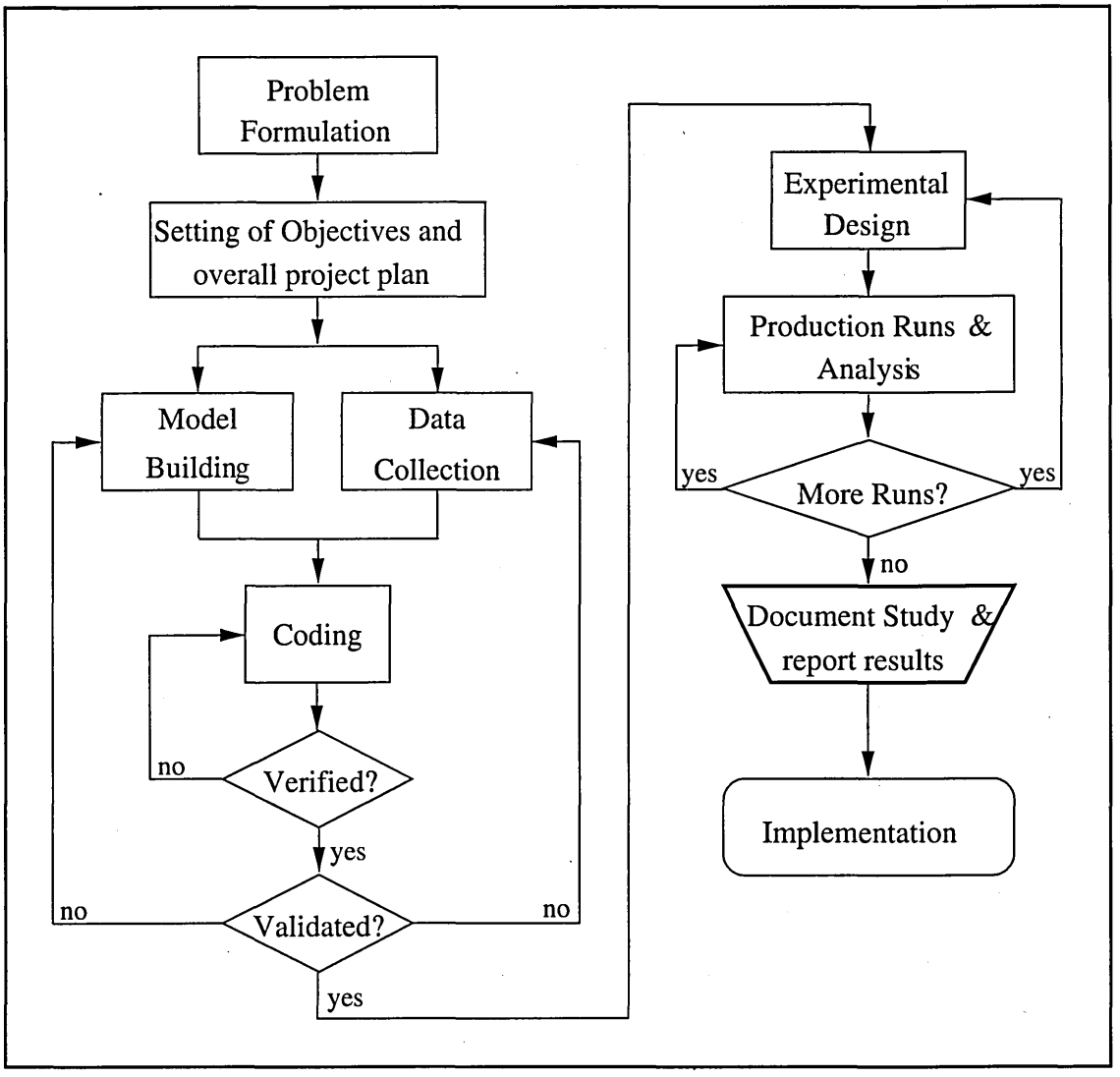

Figure 2.3: Steps in a simulation study (Banks and Carson 1984)

important if the simulation is to be reused and/or modified, especially if that will be done by another analyst

12 ***Implementation***. Depending on how well the previous steps have been conducted, the implementation of the results from the experiments will be more or less successful.

In addition to these 12 steps, Banks and Carson also broke their simulation study into 4 phases (see also Fig 2.4 on the following page):

1 Orientation. This phase consists of steps 1 and 2. Adjustments and corrections to the assumptions and objectives may occur in this phase, or a later phase. In the latter case, a restart of the process may be necessary

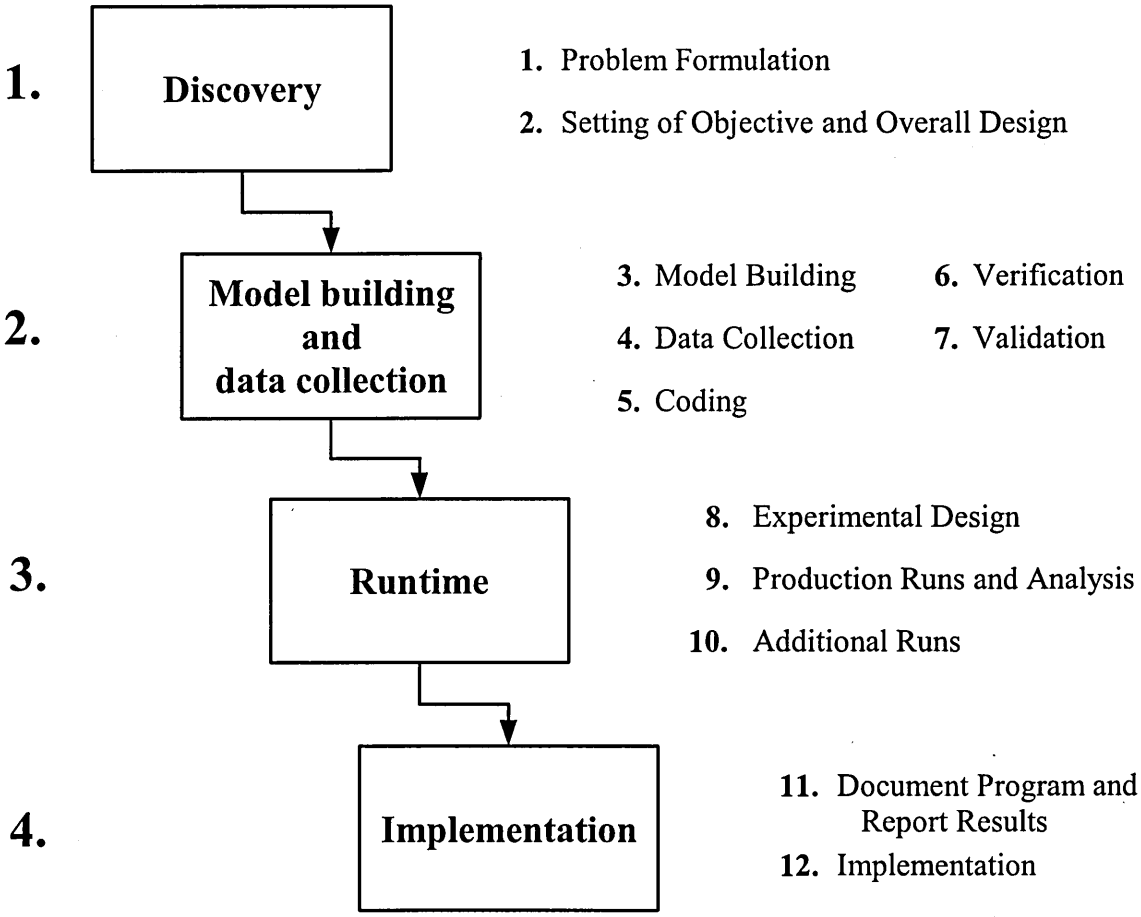

Figure 2.4: Phase structure (Banks and Carson 1984)

2 Modelling. This phase consists of steps 3 through 7 and needs constant involvement of the problem owner.

3 Running the model. Here, steps 8 through 10 are at work, delivering statistically sound results.

4 Implementation. This phase involves steps 11 and 12.

2.3.4 Kelton, Sadowski and Sadowski Kelton et al. (2002) proposed an 11-step approach:

1 Problem formulation. This consists of finding and formulating the problem and getting an understanding of the system under study.

2 Setting of objectives. Here, clear objectives for the study and its goals are established

3 Build conceptual model, deciding on the right level of detail, all of which should be backed by the client.

4 Code the model. This refers to the translation of the conceptual model into computer code

5 Model verification. Once the model is coded, logic of the model should be assessed with the problem owners and the model should also be verified as to whether *"the right things happen with 'obvious' input"* Kelton et al. (2002).

6 Model validation. Here, the focus is on verifying that the simulation do model the real system.

7 Design of experiments. Once the model is verified and validated, the next step is to plan the experiments to be run to get the right answers to the right questions.

8 Run the experiments. At this point, the experiments designed in the previous step are carried out to obtain results for later analysis.

9 Results analysis. Here, the results obtained from the experiments are analysed statistically.

10 Get insight. This means making sense of the analysed results, finding the implications of the results so they can be used for decision making.

11 Documentation. All the steps listed above should be documented, including the goals and aims of the simulation as well as the insight gained from analysing the results. This is important if the model is turned over to the problem owner for future use by someone else.

2.3.5 Benjamin et al. In their paper, Benjamin et al. (2000) proposed another concept, consisting of only three levels, as depicted in Fig 2.5.

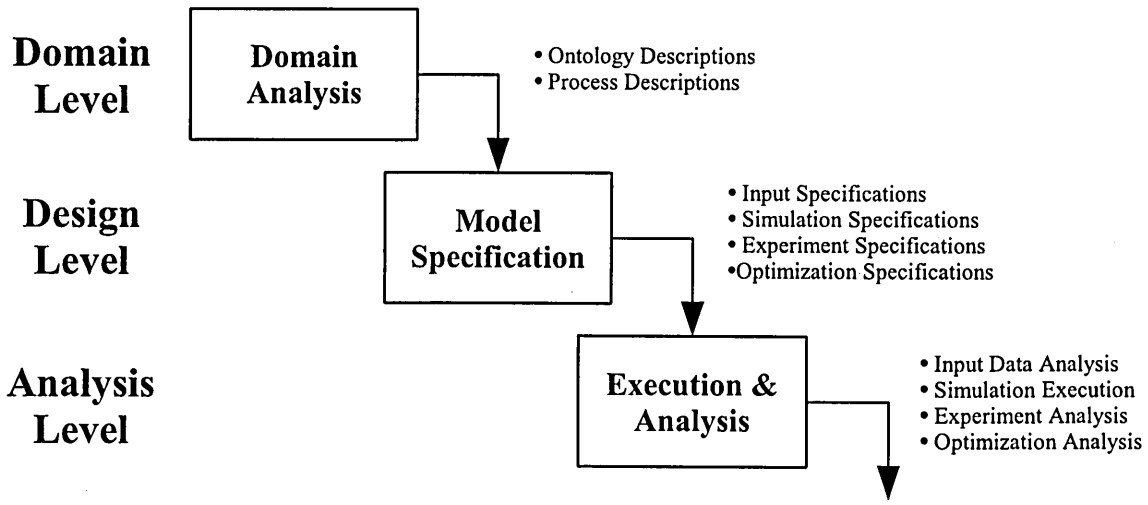

Figure 2.5: Separation of Levels (Benjamin et al. 2000)

At the domain level, information about the system and the problem to be studied are gathered. Benjamin et al. (2000) referred to this also as *"domain analysis"*.

At the design level, where "model specification" takes place, the objective and goals of the simulation project are defined, the model conceptualised and performance measures established.

The design of experiments, running them and analysing the results take place at the *"execution and analysis level"*.

Furthermore, they created this approach specifically with the aim of enabling or simplifying reuse.

2.3.6 Other approaches Robinson (1994) suggested a four-stage process, stating that:

> *"Despite the fact that these have been shown in a linear fashion, moving from problem definition to model building and so on, the additional arrows aim to demonstrate the iterative nature of the process."*

This process is depicted in figure 2.6

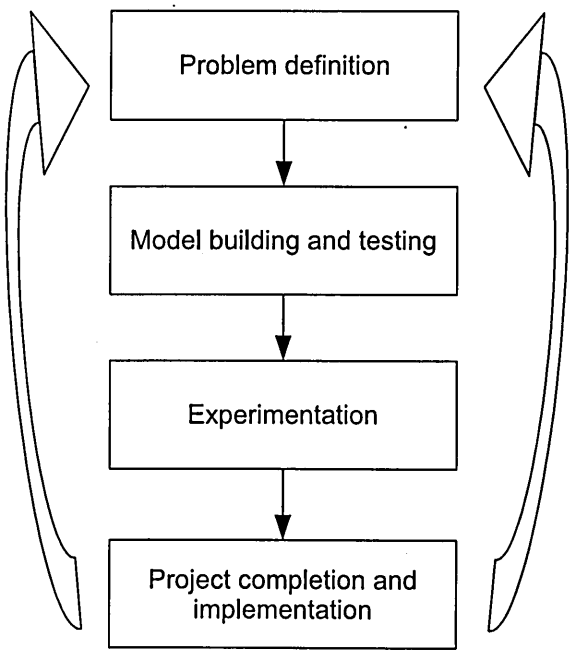

Figure 2.6: Simulation projects: an overview (Robinson 1994)

He further decomposed this process into sub-processes, consisting of three to five stages each.

Tye (1999) developed a methodology where verification and validation are done concurrently, together with the specification and design & development stages.

Robertson and Perera (2001) addressed the problems with data collection in great detail, looking at the issues of:

- data accuracy
- types of data sources
- data systems
- data duplication
- timeliness

They then went on to suggest four data input methodologies offering increasing automation for the read/write processes that link the simulation to the input data.

The first methodology is based on the input data being directly input to the simulation model by the model builder. In the second methodology, the input data is read out from a intermediary computer application, such as a spreadsheet or database which needs to be manually populated by the model builder/project team. Their third methodology consists of an intermediary software that automatically reads and writes the relevant data to and from the corporate business system, e.g. ERP (Enterprise Resource Planning) systems. The fourth methodology put forward by Robertson and Perera consists of an automated read and write solution that directly links the simulation model to the corporate business system, thus eliminating the intermediary computer applications.

However, they also pointed out that the latter two methodologies, although in principle reducing the complexity of data collection, also carried certain problems. Among them is the difficulty to create an interface for the ERP system that provides the right data in the right amount. Another is the fact that there is no industry standard for ERP.

De Vreede et al. (2003) described a modelling method in which they differentiated between the *conceptual model*, which describes *"the structure of the organizational processes and their coordination"*, and the *empirical model*, which contains more detail than the conceptual model it is derived from. Their conceptual model contains the aspects of *network model* showing the communications and interactions between the process nodes, the *process model* showing the sequence of activities in a process and the *actor model* showing the sequence of activities of a single actor.

2.3.7 Comparison of approaches All of the above approaches to simulation modelling consist of 10 to twelve steps, which can also be grouped into a few phases, as Banks and Carson (1984) and Benjamin et al. (2000) showed. Another important point that threads through all those approaches is iteration. Though this principle is most obvious when it comes to verification and validation of a model, it also has to be noted that a change at every single stage is possible. As Banks and Carson (1984) explicitly stated, *"there are occasions where the problem must be reformulated as the study progresses"*; *"there is constant interplay between the construction of the model and the collection of the needed input data"*; *"as the complexity of the model changes, the required data elements may also change"*.

The above statements are important as they show that change can happen in any phase, at any stage in the modelling process. Thus, this forms the basis for investigating which changes are most difficult to implement or most frequent.

The data collection issue treated by Robertson and Perera (2001) does not constitute an approach in itself, but points to another important element that can aid in making simulation modelling more adaptable, and changes easier to implement.

For the remainder of the work, the *steps in a simulation study* proposed by Law and Kelton (2000) was chosen, as this approach offered the best starting point for including the steps necessary to make simulation models more adaptable.

2.4 Types of adaptability in Simulation Models

> *"Flexibility is the ability to respond effectively to changing circumstances."*
> Mandelbaum (1978)
>
> Adapt: *"1) fit, adjust, make suitable 2) alter or modify to fit for a new use, new conditions 3) undergo modification to fit a new use, new conditions"*
> The New Shorter Oxford English Dictionary.

Before looking at the different approaches devised to improve the adaptability of simulation tools, a quick overview of different definitions in the area of flexibility/adaptability is considered useful.

Herrmann et al. (2000) quoted Mandelbaum (1978), who distinguished between action flexibility and state flexibility, and Buzacott (1982), who differentiated between job

and machine flexibility.

By **action flexibility**, Herrmann et al. referred to planning, e.g. a new plant, without knowing the future. Mandelbaum (1978) defined this type of flexibility as *"the capacity for taking new action to meet new circumstances"* and **state flexibility** as *"the capacity to continue functioning effectively despite change"*. This refers to the ability to automatically adapt to changes, while the former requires manual intervention.

Buzacott (1982) defined **job flexibility** as *"the ability of the system to cope with changes in the jobs to be processed by the system"* and **machine flexibility** as *"the ability of the system to cope with changes/disturbances at machines/workstations"*. Both refer to the ability to process a variety of jobs, with job flexibility being focused on the system as a whole, while machine flexibility refers to the individual machine.

Slack (1983) suggested the following types of flexibility which apply to a complete manufacturing system:

- **New product flexibility**: ability to introduce a new product
- **Product mix flexibility**: ability to produce a certain combination of products
- **Quality flexibility**: ability to change the *"quality level of one or more of its products"* (Slack, 1983)
- **Volume flexibility**: ability to vary the aggregate output of a production system
- **Delivery flexibility**: ability to vary its delivery time, i.e. its production time

Slack (1991) also introduced two dimensions of flexibility: **Range flexibility** and **response flexibility**. Range flexibility looks at the range of possibilities in terms of product, mix, volume and delivery flexibility. It constitutes the time-independent capacity for adaptation. Response flexibility looks at the time necessary to bring about the modifications necessary as laid out in the range flexibility.

Furthermore, Slack (1983) differentiated between different levels of flexibility. He uses the term "flexibility" when talking about well defined limits, enabling it to be measured and "adaptability" when the overall flexibility is not readily measurable. (see Fig. 2.7 on the next page)

Gerwin and Kolodny (1992) proposed six different types of flexibility:

- **Mix flexibility**: this looks at the ability to vary product combinations; this is equivalent to Slack's (1983) product mix flexibility
- **Changeover flexibility**: This aspect of flexibility looks at the ability of introducing a new product, i.e. at product innovation

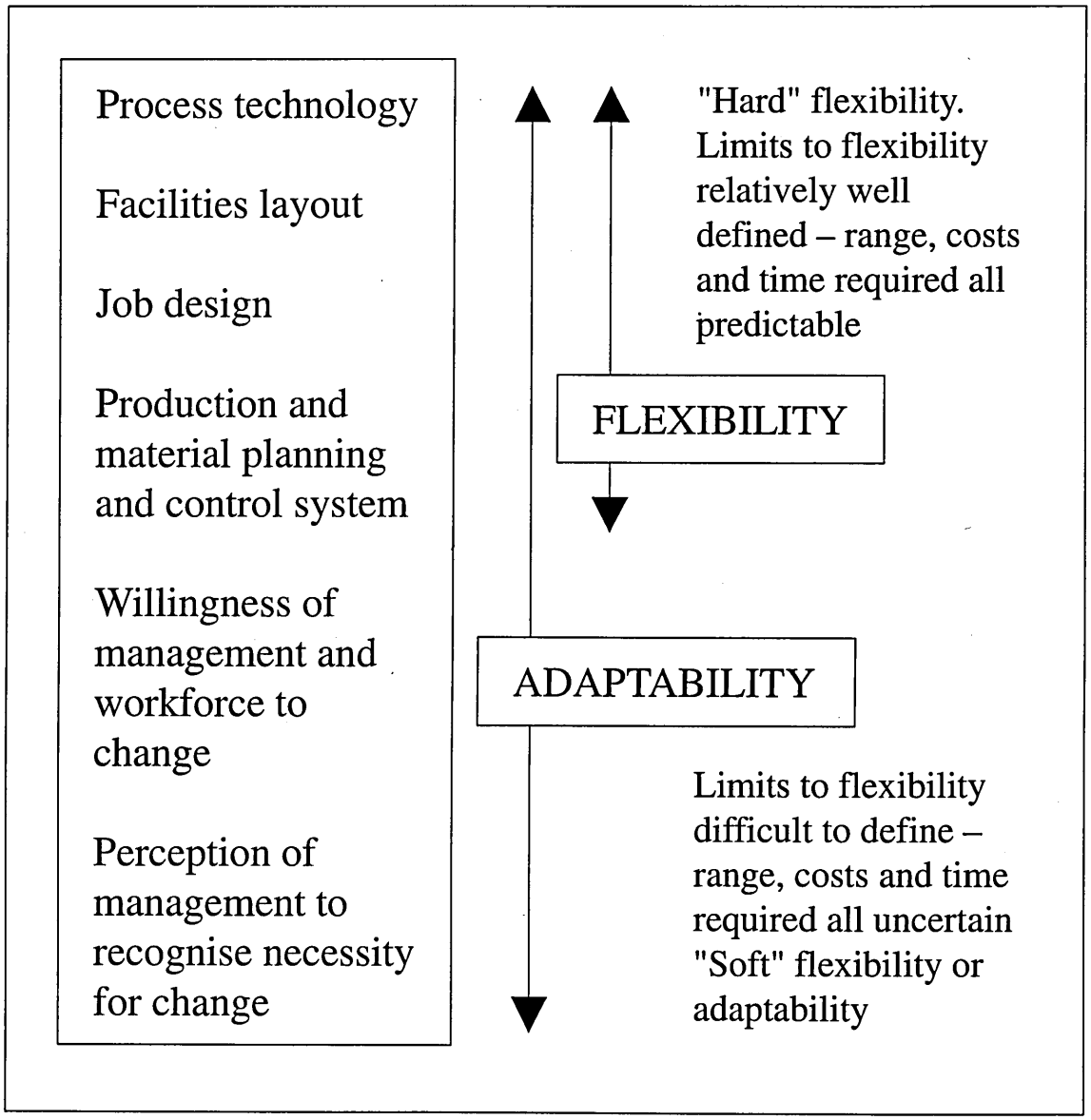

Figure 2.7: Flexibility factors (Slack 1983)

- **Modification flexibility**: ability to customize product attributes on an *ad hoc* basis
- **Volume flexibility**: enables changes to the aggregate output; this is equivalent to Slack's (1983) definition of volume flexibility
- **Rerouting flexibility**: *"the ability to adjust the sequence of machines through which a part flows"* (Gerwin and Kolodny, 1992). This deals with long- and short-term downtime aspects
- **Material flexibility**: ability to deal with variations in composition of materials and in dimensions

In their classification, mix, changeover, modification and volume flexibility are market oriented. This is because the demand for products is the relevant uncertainty. Re-routing and material flexibility are process operations oriented, as the relevant uncertainties reside within the manufacturing technology and/or its inputs.

Weihua and Baofeng (1999) identified four different classifications of flexibility: horizontal, vertical, temporal and by object of variation. They point out that in addition to these four classifications, there are also combinations of these logics, of which the most common appears to be that of *temporal* and *object of variation*.

The horizontal classification, or 'classification by phases', looks at the value chain. That is, it looks at the flexibility of the various stages of the manufacturing process on the one hand, and at the upstream and downstream phases (i.e. purchasing and distribution flexibility). They also use the terms internal and external flexibility, respectively.

The vertical or 'hierarchical' classification looks at the level of detail, e.g. the flexibility of individual resources or of the whole system. Gerwin (1987) also defined levels of flexibility to help with the classification of flexibility:

1 individual machine or manufacturing system
2 manufacturing function (e.g. cutting or assembling)
3 manufacturing process for a product or group of similar products
4 the factory
5 a company's factory system

The temporal classification is concerned with the short, medium and long-term flexibility, while the classification by object of variation looks at machine, product, process, operation, routing, volume, expansion and production flexibilities.

On the other hand, Ku (1995) warned that *"flexibility is evasive because it is a **potential**, which depends on what happens in the **future**"*. She goes on to explain that, because

of the uncertainty connected to the future, the *"value of flexibility is difficult, if not impossible, to ascertain"*. Similarly to Jones and Ostroy (1984), who state that flexibility increases with the size of sets of future positions at various cost levels, she mentions that *"flexibility is associated with the initial state but measured by the number of states it can move to or the number of choices available in the second stage"*. More generally, this refers to the definitional elements of flexibility, which she suggests as:

- range: the ability to adopt different states (Slack 1983)
- time
- change
- conditions of uncertainty: the importance or cost of uncertainty
- favourability: *"the value or benefits of change"* (Ku 1995)

2.5 Adaptability of Simulation Tools

The need for adaptability, i.e. the need to implement change, has been recognized as noted in the previous section. In order to provide for adaptability in simulation models, different approaches have been devised, of which some are described here.

2.5.1 Data-driven simulators Pidd (1992) looked at data-driven simulators in closer detail, suggesting as its main features:

- *"Pre-Programmed simulation 'model'*
- *'Model' suited to range of applications*
- *No programming by the user*
- *User provides data to simulator*
- *Data numerical, logical or textual"*

Pidd (1992) defined data-driven simulators as follows:

> *"The idea of a data-driven simulator is that the user should be unaware of the code of the underlying simulation program other than that which is hinted at by its user interface."*

He then differentiated between general and domain specific data-driven simulators. Such simulators can have flow-diagram input, as is the case with GPSS (General Purpose Simulation Software), or GUI (Graphical User Interface) input, as is the case with Witness™ (or Arena™). Pidd defined as domain-specific simulators, programs that are

geared towards a narrower application, citing flexible manufacturing systems as one such example.

Pidd (1992) further explains the main elements of such a data-driven simulator, consisting of:

- simulation logic: the computer based representation/translation of the conceptual model
- libraries:
  - general library containing general routines such as scheduling events
  - sampling routines, which handle distribution generation
  - graphic library for animation and display
- model configurator: the 'interface' through which the user can input the data, whether through text or GUI
- filer, which handles the storage of (output) data
- experimental frame, in which general run parameters (e.g. replication length or number of replications) are set
- report generator, which outputs the data obtained during the simulation runs for easier analysis

2.5.2 Programming versus assembling Simulation models can be created through the use of general-purpose programming languages, e.g. FORTRAN or C/C++, special purpose simulation languages, e.g. GPSS, SIMSCRIPT, SIMAN, or high-level simulation packages such as Arena™ or Witness™ (Law and Kelton 2000, Kelton et al. 2002). As Law and Kelton (2000) pointed out, general purpose languages, such as FORTRAN, are *"highly customisable and flexible, but also painfully tedious and error-prone since models [have] to be coded pretty much from scratch every time."*. Pidd (1992) referred to packages like Witness, where the code remains hidden, as data driven generic simulators. Special purpose simulation languages, which came later, are specifically geared towards the building of simulation models, thus simplifying some tasks. They tend to include random-variate generators and built-in statistics gathering routines (Banks and Carson 1984). Application-oriented simulation packages facilitate model-building by providing a graphical user interface (GUI) and 2– or 3-dimensional animation capabilities.

While models in a general purpose language have to be coded line by line, and the model builder therefore needs to have in-depth knowledge of the programming language, this task is rendered somewhat easier with special purpose simulation languages, as these

already contain some helpful constructs. However, the model builder still needs very good knowledge of the program and programming skills with it. Simulation packages, on the other hand, offer the greatest ease in terms of model building, as the builder does generally not need to have any knowledge of the underlying code, as the constructs are represented by icons which only need to be pulled onto the modelling area.

While the ease of building simulation models increases with the use of application-oriented simulation packages, their functionality is somewhat limited. This is due to them being geared towards rather more specific application areas. As Kelton et al. (2002) noted: *"a simulation package that relies on a* fixed *number of modelling constructs with no capability to do some kind of programming in any manner is bound to be inadequate for certain systems encountered in practice."* Thus, in order to increase the functionality of such simulators, the model builder can access a lower level, i.e. the code itself to modify it. Also, these simulation packages are increasingly integrated with other software. Arena™, for example, offers extensions to integrate other programs such as Microsoft (MS) Excel and Visual Basic, as well as providing the option of using external code in the form of the *C* programming language.

Although the functionality of simulators can be increased this way, it necessitates yet again programming skills by the simulation modeller.

2.5.3 Programming differences Another differentiation in terms of simulation software distinguishes between object-oriented and component-based programming.

Joines and Roberts (1998) described simulation programs as procedural when a problem is subdivided into procedures which are either represented by components such as queues or by means of programming code with data structures. Examples for such procedural approaches are GPSS (General Purpose Simulation System), SLAM (Simulation Language for Alternative Modelling, the basis for AweSim) and SIMAN (SIMulation ANalysis, the basis for Arena™)

They warned of the following problems with this procedural approach:

- Procedures only correspond to methods and algorithms, not to real world components. This means that a context must be given for the procedures, such as queuing situations.
- Procedural simulation programs lack extensibility, which means that none of its basic processes can be altered by the end user.

- The only way to modify such simulations is through user code, compiled in a general programming language. The communication between the user code and the simulation vendor's code is through function calls, ie where the externally programmed routine or program is called and started from within the simulation code. Here, the problem is that this leaves the simulation code itself quite vulnerable to mistakes made by the user.

Object-oriented programs, on the other hand, involve the notion of encapsulation. This means that the properties of the object are included within the object itself, instead of being spread, e.g. throughout a program. This means that implementing changes to the object is quite forward. Also, objects are created in the form of object classes, which define certain general attributes, ie some sort of 'template', which can then be used as 'instances', ie copies, of this object class. Each of these copies would have the same properties as its class, but allow certain values of it to be changed. Another important aspect of object-oriented programming is inheritance, meaning that the properties of an object can be reused in another object, including additional attributes. For this reason, Callan (1994) pointed out that object-oriented programming has a great potential for reusing components. Moreover, he talked about *"an expectation that object-orientation will deliver increased productivity through the definition of reusable components and will also deliver easier to maintain systems"* (Callan 1994).

Also, designers of objects or object classes can determine which properties should be visible to the user and which should not. In procedural style, the programmer must decide whether to make the whole code available to the end user or none at all.

Joines and Roberts (1998) mentioned that programs such as Arena™ and AweSim offer extended possibilities, called object-based features, where new objects can be created out of existing objects. However, apart from objects through composition, these programs do not offer any more possibilities. This means that new objects, which are completely independent from existing basic building blocks, cannot be built.

2.5.4 Generic simulators McLean et al. (2002), McLean and Leong (2002) and Lee et al. (2003) proposed a software architecture with standard data interfaces. Their approach was aimed at machine shops, with the architecture to enable rapid reconfiguration of machine shop simulators. Their proposal is based on a National Institute of Standards and Technology (NIST) program in *Manufacturing Simulation and Visualization* (MS&V), established in 1999 (McLean and Leong 2001). The aim of this program

is to improve the interoperability and accessibility of MS&V for US industry.

This architecture consists of a conceptual model of a generic job shop specified in the Unified Modelling Language (UML). They chose UML as this *"is a recognized standard for structured and object-oriented modelling"* (McLean et al. 2002). UML is a visual language, in which boxes and lines are used to depict constructs. Diagrams were then created for use cases, static data structures, sequences and states/activities. Within the use case diagrams, various generic forms were built to represent the various activities and actors that can occur. The UML models are then translated into Extensible Markup Language (XML). The XML files are of textual format, i.e. they do not need being compiled into binary computer code. As a result, XML files can be read by humans and can be easily implemented in off-the-shelf software applications without the need for special, additional programs.

Apart from the UML/XML structure, the architecture proposed by McLean et al. (2002), McLean and Leong (2002), Lee et al. (2003) consists of a machine shop emulator and a discrete event simulator, which are linked, and a user interface. The emulator manages the sequencing of orders, tasks and jobs, while the discrete event processor manages the flow of entities through the generic event queues and the state changes. The graphical user interface (GUI) provides for the data entry to the system. The main idea behind this approach is the linking of *"machine shop software applications with simulation"* via standard interfaces (Lee et al. 2003). The provision of such a standard interface would in turn drastically reduce the time – and costs – associated with the construction of the simulation model and the data exchange.

Their machine shop emulator contains more data than necessary to run a simulation; in fact, McLean et al. (2002) stated that *"the shop data model currently encompasses a significant portion of the data required to actually run a real machine shop"*. The idea behind this was to eliminate the abstraction phase, where data from the actual system is simplified for the creation of the simulation model.

Their machine shop emulator was implemented in Visual Basic™ (Microsoft™) and SIMAN simulation language (Rockwell Software), the GUI was created in Visual Basic alone, while Arena™ (Rockwell Software) is used as the discrete event simulator.

Son et al. (2003) developed neutral model libraries from which simulation models for various simulation packages can be built. Their approach consists of neutral model libraries, a database and a model builder. The libraries were created through *EXPRESS*,

an ISO standard to describe language-neutral information models and *EXPRESS-G*, a graphics-based version of *EXPRESS*. Based on those models, a Microsoft Access database was created, containing all the components from the EXPRESS models and the relationships between them. The actual simulation model was then built through the use of a model builder, which they created for Arena™ and ProModel™. Their model builder gathered the necessary information from the Microsoft Access™ database. This approach is based on the NIST initiative of libraries for formal, neutral models of simulation components. Son et al. (2003) stated that *"Each of these components would have views tailored to specific modeling scenarios ... defined by different modeling templates – such as an equipment simulation, a material flow simulation, a supply chain simulation and so forth"*. Furthermore, they have oriented their development towards internet-based simulation.

2.5.5 Adaptable simulation models Gahagan and Herrmann (2001) noted that simulation models are not used to their full capacity *"due to the cost of maintaining an accurate simulation model"*. They defined this maintenance cost as the *"man-hours necessary to update a simulation model"*.

They next proposed to ways to reduce this maintenance cost:

- reducing the maintenance cost by making the model more adaptable
- eliminating the maintenance cost by automating it, thus eliminating the need for respective man-hours

More generally, McLean and Leong (2001) suggested that cost can affect the very basic decision of whether to use simulation at all. Some of the factors they consider of prime importance in that respect are:

- The company's resources
  - availability of discretionary funds
  - simulation skills and experience base of current staff or consultants
  - existing information systems infrastructure (availability of required computer systems, related software applications, and data bases)
- scope and complexity of the target simulation application area
- availability of turnkey or readily-adaptable simulation models and solutions
- availability and format of input data
- cost and risks of implementing manufacturing systems without the use of simulation
- time-related costs

- salaries
- training classes, learning curves and maintenance
- translation of existing data
- systems integration with other software applications

The approach by Gahagan and Herrmann (2001) looked at adaptability of simulation models from the point of view of a production control framework, more precisely in the form of push/pull categories. In their framework, queues, workstations and the shop interact with each other and the outside world via controllers. Those queue, workstation and shop controllers then operate based on 4 component types and their attributes. In their approach, components can be physical elements as well as information, and are grouped into four types, which are further specified by attributes, such as queue entry time, the destination workstation etc. The 4 basic component types are categorized as follows:

- Type 1, Material components
- Type 2, Product permission components
- Type 3, Resource permission components
- Type 4, Batch components

This framework was implemented as an Arena Application Solution Template (AST) in the Arena Professional Edition.

Oscarsson and Moris (2002) also looked at adaptability of simulation models. However, rather than focussing on the control framework, they investigated the modification a manufacturing system goes through during its life cycle, which is much longer than that of any individual product produced in with this system. More specifically, they state that "*[d]uring its life the manufacturing system will go through a number of changes*".

As reasons for such changes to the manufacturing system they state:

- *Introduction of a new product*
- *Rationalisation of the product*
- *Changed legal provisions*

Furthermore, they stated that the changes affecting manufacturing systems are relatively infrequent and irregular. They also point out that as a consequence of those rather infrequent and irregular changes to the system itself, any simulation of such a system might have to be modified by a different staff member than the original creator of the model. This, in turn, results in the need for staff other than the original modeller, to understand the model to the point that they can modify, update and reuse it.

One way to improve understanding of such models, according to Oscarsson and Moris (2002), is to use standardized notations for the documentation, such as *Unified Modelling Language* (UML), *Integrated DEFinition language 0* (IDEF 0) and *Jackson Structured Programming* (JSP).

They identified 6 criteria that need to be satisfied in order to obtain a good documentation:

- **Neutral notation:** not limited to specific languages, systems or software, but able to support a variety of simulation packages.
- **Generic notation:** to describe a variety of systems of different purpose, complexity and scope
- **A recognised notation:** to improve communication and prevent misunderstandings and interpretation difficulties
- **User friendly notation:** to help readers overcome issues relating to the difference between natural language and abstract code
- **Descriptive in several levels:** to allow different types of users to gain access to a description suitable to their needs, e.g. bottom-level for the modeller/programmer, higher level for a model user
- **In-house competence:** use of documentation standards the company is familiar with (e.g. IDEF0 or UML)

2.5.6 Other approaches Another approach to make simulation models more adaptable is the High Level Architecture (HLA). This approach, which is widely used in military simulation applications, was created in order to combine existing simulations into one. As various simulations are built on different packages, or written in different languages, the problem of incompatibility had to be overcome. In the case of HLA, this was done through 'wrapping' the existing application into a single virtual object, a 'federate'. This encapsulating is done through an Object Model Template (OMT). The combination of federates into a larger simulation is called 'federation' (Davis and Moeller 1999).

Yilmaz and Ören (2004) stated that "*a model is reusable to the extent its original assumptions are consistent with the relative constraints of the new simulation study*". They also pointed out that, for model reuse to be successful, all dependencies and assumptions must be made explicit. Yilmaz and Ören (2004) then specified the different dependencies that exist between models and modelling context, as well as among models, and emphasise the importance of separating the conceptual model from specific simulation

components.

Spiegel et al. (2005) were concerned with the reusability in itself, proving that it is very challenging to gather all assumptions and constraints. They then suggested that for models to be composable and reusable, "*comprehensive identification of constraints*" would be necessary, and that methods for identifying critical constraints need to be developed.

Zhao and Verbraeck (2005) proposed a framework that allows multiple users with differing needs in terms of modelling detail to make use of a hierarchical simulation system. Their framework aims at supplying users at various hierarchical levels within a company with a web-based simulation tool that works at different levels of model detail, depending on the user's needs.

Garlan et al. (1995) previously proved the issue of incompatibilities, and lack of knowledge of assumptions and pointed out four aspects that need to be satisfied to achieve long-term compatibility and reusability:

1 *Explicitness of assumptions*, ie the need to explicitly document all assumptions, so as to make it easier to detect mismatches.
2 *Use of orthogonal sub-components for constructing large pieces of software*; build subsystems that can work independently of their higher-level system.
3 *Provision of techniques for bridging mismatches*, ie provision of tools to aid with wrapping components and data translation to overcome those problems.
4 *Development of design guidelines*, ie of generally applicable rules for the development of components to simplify their reuse.

2.5.7 Discussion of adaptability approaches As has been argued in § 2.5.2, programming languages are highly customizable and adaptable, but they need the most profound programming knowledge to be used to their full extent. At the other side of the spectrum, the commercial off-the-shelf (COTS) simulation packages greatly reduce the necessity of programming knowledge, but simultaneously limit their adaptability.

In comparison with the procedural approach, the object-oriented (o-o) approach may, to some extent, simplify the programming process and make it easier for simulation users knowledgeable in o-o programming to create new instances or classes. However, this approach does not help the average user of a COTS simulation package, who has no, or only limited, programming knowledge. Furthermore, the majority of simulation packages

is still procedural and will probably remain so in the near to mid-term future because of the reluctance of companies to retrain their staff on a new system and recreate existing simulations in the new package.

The *object-based features* mentioned by Joines and Roberts (1998) extend the use of procedural software, but still requires programming knowledge and does not offer the same programming adaptability as o-o programming does.

Overall, the approaches taken by McLean et al. (2002) and Son et al. (2003) aimed at simplifying the generation of simulation models, enabling the reuse of existing ones and the construction of complex models from simpler ones. Son's approach further aims at improving internet-based simulation facilities. Hence, this approach constitutes a novel way to improve and increase the use of simulation in the manufacturing domain.

The concept of interoperability in simulation is not new (Banks 1997; 2000, Pidd et al. 1999). However, apart from first tentatives in research, there is no sign of a wider-spread introduction of such standard data formats. More importantly, the idea of the data model containing *"a significant portion of the data required to actually run a real machine shop, not just simulate its operation"* threatens to increase the amount of data to levels which are hard to oversee. This, in turn, could jeopardize any attempt to manually modify an existing model based on this approach. Furthermore, this also poses a problem in terms of simplifying the data in order to obtain a valid, yet simple model. Specifically, Lee et al. (2003) warned that the problem of abstractions of real-world data for simulations is not well addressed in their current model. Furthermore, while this approach may simplify the generation of simulation models and its maintenance, it is restricted in so far as it only looks at generic simulators. Also, the first prototype has been created for machine shops – other sectors would therefore need yet more generic simulators. Additionally, it remains to be seen to what extent a generic simulator fits other companies in the same domain, or whether this will lead to the same problems so far only associated with *"user templates"*.

The approach by Son et al. (2003) presented an interesting alternative. The neutral information models, combined with translators for various simulation software packages, would undoubtedly simplify the reuse of simulation models, as the neutral model could be imported to various packages through translators. However, this approach is more complicated in other respects: Instead of building a simulation model in the simulation package the user is familiar with, the model needs to be built in a neutral format before the model builder translates it to the familiar package. Also, the need for pre-defined

modelling templates further limits the adaptability of this approach.

The production control framework outlined by Gahagan and Herrmann (2001) aimed at simplifying, and thereby increasing the adaptability of, production control policies. Furthermore, their approach looks at the creation of modules or templates, which have a great potential in reducing the maintenance time in terms of changing some data. However, the creation of templates has the inconvenience that their basic structure cannot be changed, i.e. the logic itself remains static and out of reach for most users (as the Professional edition is necessary). Also, their approach only looks at a small fraction of the manufacturing domain.

While there is no arguing about the necessity and advantages of good documentation, the approach taken by Oscarsson and Moris (2002) focussed solely on that aspect of simulation studies, rather than the implementation of changes in the model.

HLA by itself, as explained by Davis and Moeller (1999), is very useful in combining independent simulation models, regardless of the programming language used. However, this is not a tool which is easy to use, nor does it, by any means, make an existing simulation model more adaptable.

The approaches suggested by Garlan et al. (1995), Yilmaz and Ören (2004), Spiegel et al. (2005) and Zhao and Verbraeck (2005) all point out the importance of knowing the assumptions, and give ideas on how to develop reusable, reconfigurable software, but do not offer any substantial practical approaches towards improving the easiness of modifying existing simulation models.

As a result of the shortcomings of the solutions presented above, the decision was taken to devise a new approach to make existing simulation models, created in widely-used simulation packages, more adaptable.

2.6 Measuring adaptability

Herrmann et al. (2000) devised a measurement technique to study the adaptability of simulation models. They defined an existing simulation model requiring certain changes as M_0 and M_1 as the new simulation model incorporating those changes. They denoted the effort required to create M_1 by changing M_0 as $E_{(M_1-M_0)}$ and $E_{(M_1)}$ as the effort required to build M_1 completely. From this, they defined the adaptability of M_0 compared to M_1

as $A_{(M_0,M_1)}$ and:

$$A_{(M_0,M_1)} = \frac{E_{(M_1-M_0)}}{E_{(M_1)}} \tag{2.6.1}$$

They further suggested *time* or *cost* as measurement for this effort. More specifically, they proposed to:

> *"measure effort by counting the number of data values that must be added, deleted, or changed, [and for] simulation models that are sets of statements, ... the number of statements that must be added, deleted or changed."*
> (Herrmann et al. 2000)

However, they also warned that *"adaptability cannot be measured independently of the change being made"*. Moreover, they stated that the particular model and the changes being modeled as well as the software package used affect the adaptability.

2.7 Surveys on simulation software

2.7.1 Previous surveys Hlupic (2000) conducted a survey of simulation software users in the UK, distinguishing between academic and industrial users. In this survey, 44.4 % of respondents used Simul8, 38.8 % used Witness™ and Siman/Cinema (the precursor to Arena™) was used by 33.3 %. 58.8 % of academic and 25 % of industrial users mentioned problems in modelling caused by software limitations and inflexibility. Also, 44.4 % of users complained about a "lack of modeling facilities/flexibility". More specifically, 33.3 % of academic users and 22.2 % of industrial users complained about limited flexibility.

A survey by Murphy and Perera (2001a;b) focused on differences in practising simulation in the US and UK. Their survey suggests various major and minor factors that contribute to successful simulation studies.

Two of these major factors are:

- creation of custom-built model input/output interfaces *"to facilitate the ease of model building for both experts and non-skilled users"* Murphy and Perera (2001a).
- separation of input data from model logic *"so the model can be updated with new data without massive complication"* Murphy and Perera (2001a).

Three of these minor factors are:

- selection of software depending on application, capabilities and future integration
- libraries of generic objects to limit recoding
- integration of simulation with business systems

Based on their findings, Murphy and Perera determined a list of 'best practices' proven to be successfull. These 'best practices' fall into of 4 categories:

1 introducing simulation,
2 establishing simulation,
3 practising simulation and
4 developing simulation

In the *sixth biennial survey of discrete-event simulation software tools*, Swain (2003) focused on various aspects of simulation packages. Those aspects include the typical applications, model building facilities (e.g. graphical construction and debugging tools), input distribution and output analysis tools. His survey of 49 different simulation software packages also looks at code reuse components, such as templates, and animation. From this survey results that the large majority of today's modelling tools (close to 82 %) use graphical model building (drag and drop) and run time debugging, and almost 86 % of the software surveyed allows access to programmed modules, i.e. the simulation code itself. The survey also showed that close to 84% of the simulation packages offer some form of code reuse possibility (e.g. templates). In terms of animation, the survey showed that 75.5 %, i.e. $^3/_4$ of software packages offer post-run as well as real-time animation.

Eldabi and Paul (1997; 2001) assessed three simulation packages, VS7, Simfactory II.5 and Siman/Cinema IV (the precursor to Arena) on their ability to model varying levels of detail.
They have done so by attributing a number ranging from 1 *(absence or poor matching)* to 3 *(excellent quality or matching)* to the three packages for a list of attributes, such as "*quick and simple model building*", "*running speed*", "*detailed model building*" etc.
Their results indicate that there is no one simulation package ideally suited for modelling systems at varying levels of detail, and that some packages are better suited for modelling at the conceptual level, while others work better for detailed modelling.

Baldwin et al. (2000) conducted a survey of European simulation specialists, looking at, among others:

1 type of software used (i.e. whether general-purpose programming languages, COTS packages etc)
2 number of packages used
3 main purpose of simulation
4 users' general opinions about the software
5 respondents' opinions about main limitations of the software packages
6 users' opinions about most important positive features of the software packages
7 respondents' opinions about features that should be included in packages

The results show that:

1 more than half (57 % of the approximately 40 respondents use a simulator (such as Arena, Witness, ProModel...), 38 % use only languages and about 5 % use both
2 64 % or respondents use only one package
3 only 5 % of respondents use simulation solely for education purposes, while 39 % use it solely for modelling real systems
4 more than 30 % believe that the flexibility of the packages they use could be improved
5 more than 35 % of respondents stated that graphics/animation is the most important positive feature of the software used
6 the top 5 requests of features for simulation packages were (in decreasing order)
    - better software compatibility
    - facility for output analysis
    - link to database
    - more flexibility/help in experimental design
    - better online help/better experimentation facilities/support for standard programming concepts.

2.7.2 Comparison of surveys The survey by Hlupic (2000), although focusing on the users' opinions, dates back to 2000, and was only addressed at British simulation users. Also, her survey did not go into the details required for the explicit study of adaptability of modern simulation software as intended by this author.

Murphy and Perera (2001a;b) found that there are large differences between users in the UK and the US and identified 'best practices' that should be followed to successfully deploy simulation. Some of those 'best practices' are specifically targeted at improving the flexibility of simulation models and software.

Swain (2003) offered insight with respect to some main functions of modern simulation software, but does not indicate any user preferences or usage of various packages.

The surveys conducted by Eldabi and Paul (1997; 2001) showed some interesting results in the domain of simulating with varying levels of detail. While this survey shows some interesting results relating to a certain type of flexibility of simulation packages, it does not indicate in any way how software users feel about the difficulty associated with certain tasks in building and modifying simulation models.

Although the survey by Hlupic (2000), with a response rate of 25 % (out of 220 questionnaires), was a good starting point, it only looked at British simulation users. The one by Baldwin et al. (2000) was an improvement on that, as it looked at European simulation users, but the response rate was quite low (only 30 % of 120 questionnaires were returned). Also, neither of these surveys looked at the type of tasks most difficult to implement/change, nor was the sample truly representative of the world wide descrete event simulation sector. It was thus deemed necessary to conduct a new survey of simulation software users, on an international basis, in order to obtain results relevant to the adaptability of simulation software and the basic tasks associated with the modification of simulation models.

2.8 Key findings

As indicated in section 2.3 on page 7, all approaches to building simulation models consist of 10 to 12 steps, although those steps can also be grouped into a few, more general, phases.

While all these approaches implicitly allow for change, none of them seems to look at what types of changes are most frequent, or most likely to affect the later modification of an existing, working but shelved, simulation model.

From the various definitions of types of flexibilities in section 2.4, it emanates that, due to its orientation towards the future, it is intrinsically evasive, for the future cannot be predicted; hence the definition of flexibility by Ku (1995) as a *potential*.

The section on the adaptability of simulation tools (section 2.5) shows that there is a

substantial number of different approaches, of which some are more flexible (e.g. object-oriented programming, high level architecture) than others, while some are easier to handle than others (e.g. high-level simulation packages such as Arena, Simul8 or Witness being much easier to handle than object-oriented programs). There are also some promising attempts for generic simulators. Overall, however, section 2.5 shows that there is little available that would help make existing simulation models, created in widely used simulation-packages, more adaptable.

While the measurement technique for the adaptability of simulation models devised by Herrmann et al. (2000), as described in section 2.6, is perfectly suitable for the aims and goals of this thesis, the surveys looked at in section 2.7 are somewhat inadequate. While all surveys are helpful in gaining an insight into user preferences and proficiencies in certain geographical areas, it was deemed necessary to conduct a new survey of simulation software users on an international basis to gain a wider picture, and more detailed answers relating to the adaptability side of simulation. This is due to the fact that none of the previously obtained results focused specifically on the difficulty in implementing the changes, as was considered necessary for this thesis.

Chapter 3

Research Methodology

This chapter describes the different approaches used to answer the research questions. It outlines the different methodologies and the reasons for using various approaches in research.

3.1 Key Stages

The research on a new methodololy for the design of adaptable simulation models consists of three main stages, hereafter referred to as *Research Stage I*, *Research Stage II* and *Research Stage III*.

The first stage consisted mainly of defining research methodologies, conducting the stage 1 interviews and the simulation user survey, both of which led up to the benchmarking exercise. The user surveys cited in the literature review pointed out some elements of interest, such as the general perceived lack of flexibility/adaptability, they did not give a clear picture of the tasks easiest, or most difficult, to modify. In order to counter that lack of data, this author conducted a new user survey. Another issue countered by this survey was the lack of world-wide simulation users' opinion, as all results stated in the literature review were restricted to a much smaller geographical area. The interviews were considered necessary to get some initial data and ideas for the construction of the questionnaire.

The benchmarking exercise makes up *Research Stage II*, and consisted, on the one hand, of comparing two very popular, commercial off-the-shelf simulation (COTS) packages, and on the other hand of a small controlled experiment to find out about the influence of the use of submodels on the overall adaptability of such COTS packages.

While the questionnaire results showed that COTS packages were the most widely used, they were not sufficient for explaining why one package had such a great market share. In order to test whether there was any substantial differences, and in order to decide on one package for the validation (see section 7.2), a benchmarking of some of the most widely used packages had to be conducted. The experiment was used to test the suitability of some elements for the new methodology for adaptable simulation models.

Research Stage III consisted of developing a new methodology for the design of adaptable simulation models and validating it via interviews and a framed field experiment. While the existing methodologies, outlined in chapter 2, give excellent guidelines for the building of simulation models, they are mainly concerned with one-off model building, or the creation of new interfaces and new software to improve adaptability, but do not suggest any possibilities for creating simulation models, in existing packages, that would enable later reuse thereof under modified circumstances.

See figure 3.1 on page 40 for the key stages of this research.

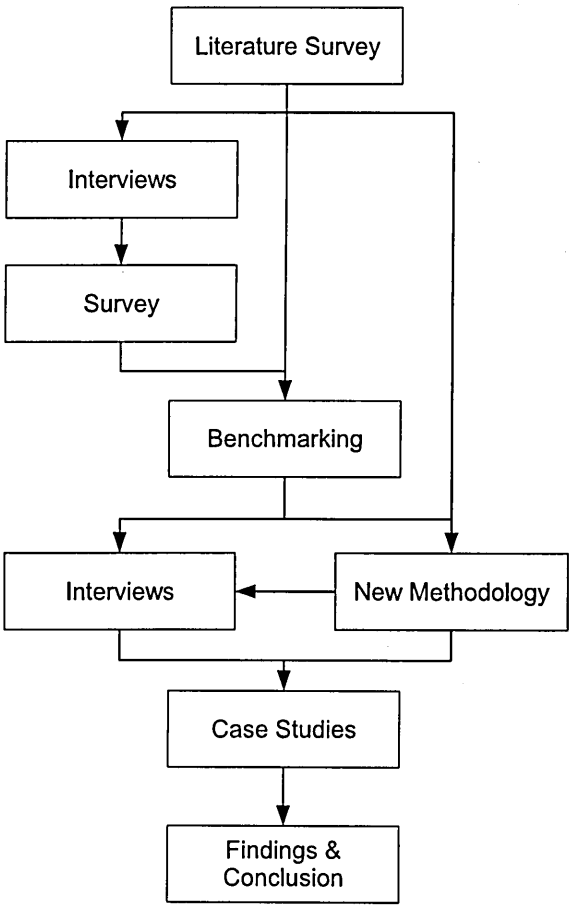

Figure 3.1: Key stages – flowchart

3.2 Interviews/ Questionnaire survey

Different settings for interviews and the according types and stages of interviews were looked at. Types of interviews can be job interviews, opinion polls and surveys, among others, while the stages of interviews consist of the opening, developing the main themes and the conclusion. Each of these phases can be further sub-divided. Also, the possible styles (e.g. standardized versus non-standardized) and the different types of questions that can occur in an interview (open-ended questions, multiple choice, ranking and probing questions) were looked at (Blaxter et al. 2001, Bryman 1988, Gorden 1975, Head 1999, Keats 2000). The issue of bias and the sequencing of questions also had to be considered.

3.2.1 Interviews *"A feature of interviewing ... is the opportunity it gives the interviewer to explore the reasons for a person's responses"* (Keats 2000). In this sub-chapter, the different settings for interviews and the according types will be described before detailing the stages of interviews. Also, the possible styles (e.g. standardized versus non-standardized) and the different types of questions that can occur in an interview are looked at. Finally, the problem of bias and the sequencing of questions will be described.

Types of interview

Literature differentiates between different applications and types of interview. First and foremost, however, it is important to distinguish interviews from ordinary conversation. Gorden (1975) pointed out that *"any two-way conversation involves many of the same skills and insights needed for successful interviewing. The main difference is in the central purpose of interviewing as opposed to other forms of conversation"*.

Keats (2000) grouped interviews into those which seek information without trying to change a person's behaviour and those which include some expectation of change in the person. More specifically, she differentiates between the forms of interview depicted in table 3.1 on the following page.

Due to this extensive list, only opinion polls, telephone interviews, group, guided and research interviews are described here in more detail.

Opinion polls, which are often used in market research and sales promotions, depend on large samples and on their nature so as to allow for statistical analysis.

Table 3.1: Interview types (Keats 2000)

Over-the-counter information services	Counselling services
Advice Bureaux	- counselling for students
At the bank	- marriage guidance counselling
At the insurance agency	- drug and alcohol counselling
Tourism and travel services	- trauma counselling
Opinion polls	- legal advice
Telephone interviews	Police interviews
- surveys	Welfare service interviews
- selling	Clinical interviews
At the school	- hospital interviews
Mass media interviews	- medical interviews
- on television	- paramedical interviews
- on radio	- nursing interviews
- in newspapers and magazines	The cognitive interview
Job interviews	Research interviews
- for selection	Interviews to obtain participants
- for progress evaluation	in laboratory studies
- for retrenchment	

Telephone interviewing can be used for information enquiry as well as advice-seeking, such as opinion polls or psychological counselling. As a result, different approaches are used.

In group interviews, either the interviewer or the respondent is replaced by a group. The former is often called an interviewing panel, which tends to be comprised of experts, while the latter is called a group delegation. An example for this, as stated in Keats (2000) is the tour guide informing tourists about the historical background of a building.

Guided interviews combine interviews with questionnaires, tests or tasks, with the interviewer and the respondent(s) working through the questions together. This gives the interviewer the opportunity to clarify questions the interviewee(s) might have. Keats (2000) warns, however, that too much guidance by the interviewer can introduce bias.

Research interviews are a means to gather the required data. *"Research interviews differ from counselling and clinical interviews in several important ways. First, they are not intended to be an agent of change, although participating in an intensive interview can alter a person's attitudes and later behaviour"* (Keats, 2000). It is important that the questions are free from bias and consistent from one interview to the next. Also, the respondents need to be chosen according to the research plan. The fact that such interviews are conducted orally also benefit from probing. (See section 3.2.1)

Stages of an interview

According to Keats (2000), interviews consist of three major phases: Opening, developing the main themes and conclusion.

During the opening phase, a rapport with the interviewee is established, starting with informing him about who the interviewer is and his or her role. During this phase, it is also important to declare what the outcome of the interview will be used for and what methods of asking and recording the responses will be used. If the interview will be recorded to tape (sound or video), permission to do so must be granted by the interviewee. Also, Keats advises of informing the respondent about the estimated time needed to complete the interview. It is during this phase, too, that the interviewer will obtain background information about the respondent. This should only include relevant information, and can be used to help the interviewee relax.

During the development of the main themes, i.e. the questioning itself, the topics should begin with the least threatening aspects. During this phase, the interviewer can use various forms of questions (see section 3.2.1).

The closing phase of the interview includes telling the respondent that the interview is about to end and thanking him/her for participating. Also, the respondent should not be rushed as *"rushing the conclusion suggests that the respondent is only being used for the interviewer's own purposes rather than as someone who has something important to say on the subject"* (Keats 2000). If tape recording or a video has been made, it should be played back at this point, at least partially, if the interviewee asked for this. If a further interview with the respondent will follow, a link between the two interviews should be established here.

Gorden (1975) suggested that during the definition phase, the interviewer introduce himself and name the sponsor of the study, if there is one. Next, the purpose of the interview has to be explained and, if necessary, the selection of the respondent and guaranteeing the interviewee anonymity.

According to Gorden (1975), it is important to have a well planned opening question, even for an unstructured interview. The opening question needs to be tied in well with the purpose of the interview as explained during the definition stage. The opening question may be of a broad or narrow focus, but should ask for information which is easy for the interviewee to give, so as to make the respondent feel more at ease.

Interview style

Gorden (1975) differentiated between standardized and non-standardized interviews, with each containing sub-types. The idea behind standardized interviews is to put the same questions to numerous respondents and thus gather comparable and classifiable answers. Non-standardized interviews, on the other hand, do not pose the same questions to all respondents, thereby limiting the applicability of statistical analysis.

Gorden (1975) differentiated between scheduled and non-scheduled standardized interviews. In the scheduled interview, all questions and the order in which they are to be asked by the interviewer are specified in advance. Specifically, he stated that a schedule may specify

- content of questions
- exact wording
- sequence of questions
- context to be supplied with the question
- answer categories to be used

Non-scheduled standardized interviews leave some choice pertaining to order of (some of) the questions and the wording to the interviewer as well as the possibility of some further probing. However, he also conceded that interviews may be scheduled to varying degrees.

Gorden (1975) differentiated between preparatory non-standardized interviews, which serve as preparation for standardised interviews, and independent non-standardized interviews, which have an agenda on their own.

Blaxter et al. (2001) referred to standardized interviews as structured, to non-standardised as open-ended or unstructured and as semi-structured to interviews lying between the two.

Question types

Keats (2000) differentiated between open-ended questions, multiple choice, ranking and probing questions.

She pointed out that open-ended questions, especially very general ones, tend to be used as openers. Furthermore, as they do not suggest answers or offer alternatives, they

encourage the respondent to talk freely. Specific open-ended questions, on the other hand, give a more limited range of possible answers.

Multiple choice questions provide the interviewee with a set of possible answers, which can take several forms. Such questions are generally limited to the interviewer's choices offered, as allowing the respondent to add a category would lead to rejecting the previous list. Furthermore, in oral interviews, the interviewee might not remember all the choices offered, thereby limiting the efficiency of the question. To circumvent this problem, multiple choice questions may include a scale, representing positions from extremely positive to extremely negative. They may include five, seven or even eleven choices. Keats gives the following example for such choices: "*'always, mostly, not very often, seldom, never' or 'strongly agree, agree to some extent, neither agree nor disagree, disagree somewhat, strongly disagree*" (Keats 2000). However, as she points out, respondents may opt for the middle choice, when in fact they do not want to give an opinion. To counteract this problem, an even number of choices may be introduced, thus forcing the interviewee to one side of the scale. The wording of such choices may then be *"highly important, quite important, not very important, not important at all"* (Keats 2000). Such wording can be replaced by numbers.

Ranking questions involve several alternatives, which are to be ranked from highest to lowest rank, e.g. most important to least important or most frequent to least frequent. The ranking increases from 1 for the highest rank to the lowest.

Probing questions, according to Keats (2000), can be general or specific. What both forms have in common is that they allow eliciting more detailed information, based on a further response. They can be used to rephrase a past question to clarify its meaning. More specifically, Keats gave some examples of probing questions, grouped as 'clarifying', 'seeking the next stage in a sequence', 'seeking reasons', 'checking consistency' and 'revising'. Clarifying elicits more details, more depth from the interviewee. Seeking the next stage prompts the respondent to reveal more breadth, i.e. what happened at a later stage, while revising relates back to an earlier question or answer, eliciting more detail, based on a new understanding. *"Additional information can be obtained by probing the initial responses. Reasons for the response can be explored and all questions can be responded to without loss. This gives a richness to the data, allowing many individual differences in opinions and reasoning to be uncovered"* Keats (2000).

More specifically, Gorden (1975) differentiated between silent probing, encouragement, immediate and retrospective clarification, immediate and retrospective elaboration and mutation.

Silent probing includes the use of a pause before asking the next question to encourage the respondent to give more information. Encouragement itself may consist of verbal clues, e.g. 'go on' or 'uh huh', or nodding, or facial expression. Immediate clarification elicits more detail on the topic under discussion, such as the sequence of events, while retrospective clarification is concerned with a topic discussed earlier. In immediate elaboration on the other hand, the interviewer indicates that he would like the respondent to give more information on the topic under discussion, without the constraints of clarification, i.e. without asking for specific detail. Through retrospective elaboration, the interviewee is asked to elaborate on a topic discussed earlier. Through mutation, the interviewer introduces a new topic, or leads the respondent onto another topic. Thus, mutation is of use in the constellated structure (see section3.2.1) to make the transition from one set of themed questions to another.

Bias

Keats (2000) talked about ambiguous questions and bias. She refers to ambiguity when a question is composed of two problems, ie sub questions, when it is not specified which part the interviewee should answer. Bias can occur in the wording of a question, but also in the interpretation of the answer. Biased questions of the type "you are aware of that, aren't you?" urge the respondent to answer in a specific way; in this case, the interviewee is likely to answer to the effect of "yes, I am aware of that", even if it is not so, just to fulfil the interviewer's expectations. She also warns that bias can occur if the interviewer makes judgements based on *"appearance, speech, style, age and gender"* of the respondent (Keats 2000). Bias at the interpretation stage can stem from the respondent's inconsistency, ie a contradiction to an earlier answer, non-cooperation, evasion, inaccuracy in recall, lack of verbal skills, conceptual difficulty and emotional state of the interviewee. Non-cooperation occurs when the respondent does not answer willingly or gives only superficial answers. Evasion occurs when the interviewee talks around a question instead of answering it directly, hesitates before answering or simply refuses to answer. Inaccuracy in detail may introduce bias as such inaccuracy may be used by the respondent to place himself in a more favourable light. Conceptual difficulty occurs when the interviewee has

difficulty in understanding the question. This, as well as the emotional state of the respondent, can influence the interviewee to feel sympathy or antipathy towards the respondent, which may then reflect in the interpretation of the answers. Keats (2000) further warns that bias can occur not just in the wording of a question and the interpretation of the answer, but that it might just well be introduced by the respondent through interpretation of the question.

Gorden (1975) referred to this as 'leading questions'. *"The term 'leading questions' refers to any question, including its context and answer structure, which is phrased so that it appears to the respondent that the interviewer desires or expects a certain answer"*. Gorden (1975) differentiated between context-based and emotion-based leading questions. The former can be personal or impersonal, with the personal type being centred on the point of view of a specific individual or group, while the impersonal tries to prevent the interviewee from identifying with an individual or a group when answering. The latter introduce bias through *"emotionally charged words"* (Gorden 1975).

Keats (2000) and Gorden (1975) both noted that bias, ie leading questions, may be useful in some cases. Keats talked about leading questions being used in sales to gain a customer, and Gorden (1975) also suggested that leading question should not always be avoided. Specifically, he pointed out that leading questions can be very useful in obtaining an answer from a respondent on a subject he or she would not normally talk about. In such a case, a leading question is useful as it makes the interviewee more confident to talk about it, suggesting that the interviewer is familiar with the subject or that the respondent is not abnormal if he does practice a specific activity or holds a specific view. Moreover, Gorden (1975) identified three different situations based on bias:

1 the leading question helps to obtain more valid information
2 valid information is obtained despite a leading question
3 the answer is distorted because of the leading question

Question sequence

In addition to the basic form of questions, as described in section 3.2.1, Keats (2000) also attributed quite some importance to the structure of the interview in terms of the question sequence. The sequence of questions is tied to the interview style (see section 3.2.1), as the interviewer needs to be aware of the direction in which he is leading the respondent. For scheduled standardized interviews, the question sequence is planned in detail, before

the interview takes place, while for the others, it is left open.

Keats (2000) differentiated between simple structure, chain structure, branching structure with channelling effects, sequential structure with simple feedback loops, branching structure with complex feedback loops and constellated structures.

In the simple structure (see figure 3.2), each question is independent from the previous ones; there is no personal involvement between interviewer and interviewee. She pointed out that this type of structure is *"typical of the opinion poll or fact-finding type of interview, where there is a set list of questions and a set list of respondents"* (Keats 2000).

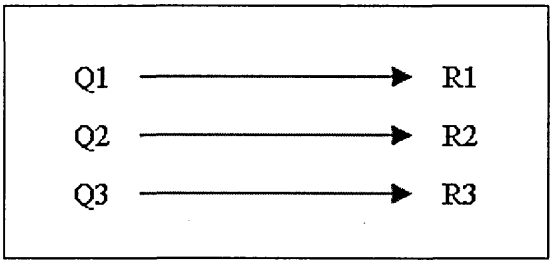

Figure 3.2: Simple structure (Keats 2000)

The chain structure (see figure 3.3) allows gaining more precise information on a specific topic, i.e. increasing the detail. Keats (2000) suggested this form of question sequence for cognitive interviews, *"where it is important to let the respondent use cognitive associations to recreate a situation in memory which may be difficult to recall"*.

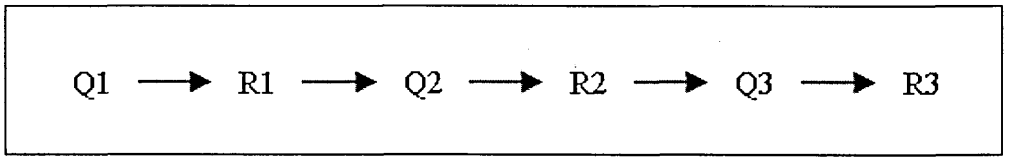

Figure 3.3: Chain structure (Keats 2000)

The branching structure with channelling effects (see figure 3.4 on the next page) is used when the answer to a question offers different aspects. In such a case, the interviewer will select one aspect and ignore others. However, it is obvious that bias is very easily introduced through this selection process. In order to overcome this bias, the sequential structure with simple feedback loops and the branching structure with complex feedback loops (see figure 3.5 on the following page and figure 3.6 on page 50) can be used. Both structures are more rewarding and less prone to bias, although they place greater demands on the interviewer, as he has to relate to earlier parts of the interview.

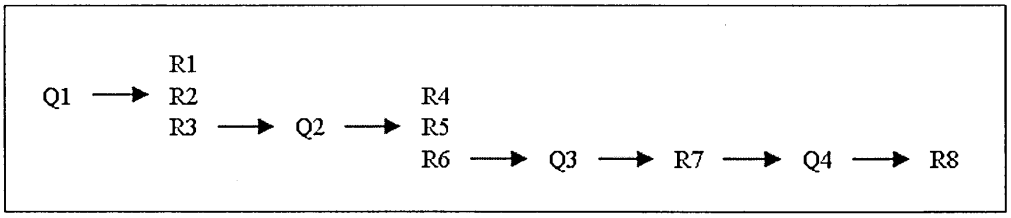

Figure 3.4: Branching with channelling (Keats 2000)

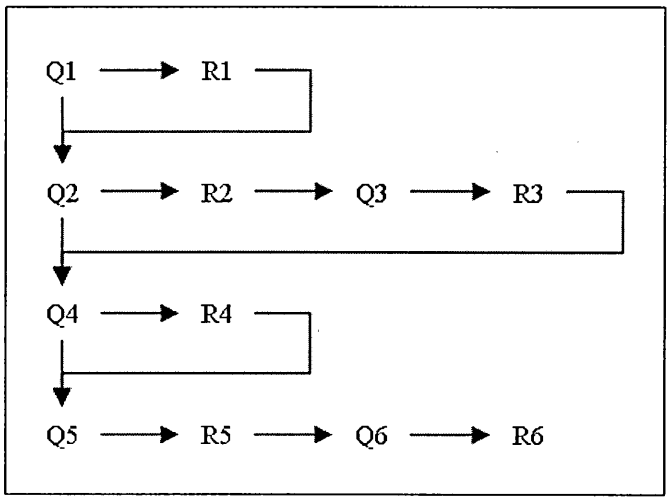

Figure 3.5: Sequential with simple feedback loops (Keats 2000)

Keats (2000) suggested that, with the exception of the branching structure with complex feedback loops, the question sequences from the other structures can also be arranged in constellated structures (see figure 3.7 on the following page), where *"five or so question-response pairs"* are combined into themed sets. When one such set is completed, the interview moves on to the next set of questions. Keats (2000) also pointed out that this constellated structure is typical for multiple-choice questions and rank-order sets.

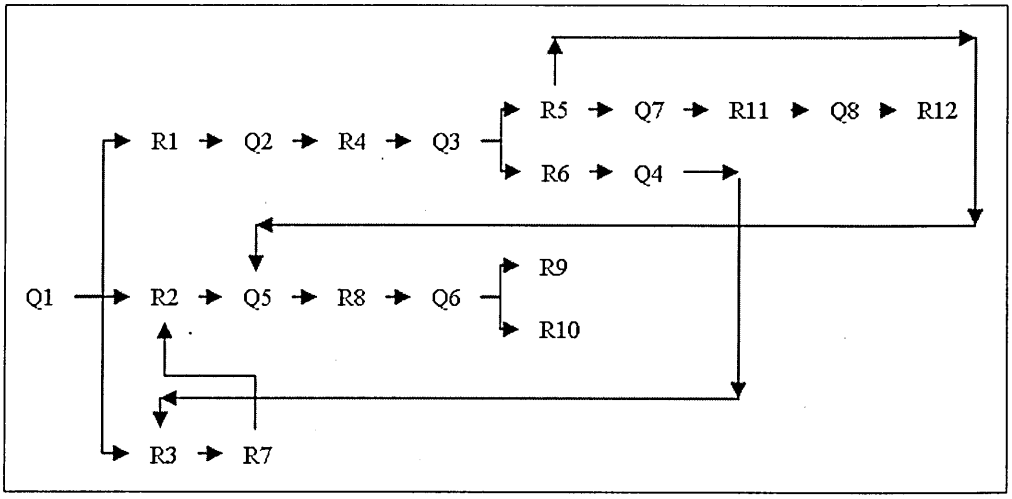

Figure 3.6: Branching with complex feedback loops (Keats 2000)

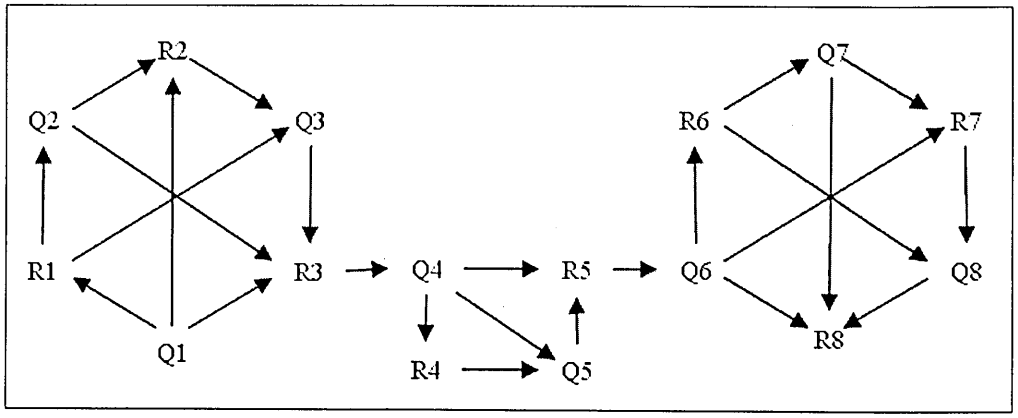

Figure 3.7: Constellated structure Keats (2000)

Quantitative versus qualitative Data

While opinion polls (see section 3.2.1) are used to gather data from a large sample, non-standardized interviews allow probing and thus obtaining very detailed, in-depth knowledge. Thus, the former is very useful for statistical, ie quantitative, analysis and the latter for qualitative analysis. In this sub-chapter, these two forms of data and their respective advantages and disadvantages are looked at.

As Blaxter et al. (2001) point out, *"quantitative research is concerned with the collection and analysis of data in numeric form. It tends to emphasize relatively large-scale and representative sets of data."* Thus, for quantitative research to take place, a large sample is necessary.

In terms of interviews, this means a large number of respondents whose answers can be clearly ranked and ordered for statistical analysis. Therefore, possible types of questions include multiple choice and ranking questions, the former perhaps aided through the use of scales. However, by only allowing a limited number of choices and confronting every interviewee with the same questions might introduce bias.

Where numerical data are gathered, such as from simulation outputs, a number of results is necessary to enable statistical validity and precision, expressed through mean value, standard deviation and confidence intervals. Obviously, the precision increases with the number of replications or values used to compute the result, which relates to a smaller standard deviation.

> *"Qualitative research is concerned with collecting and analysing information in as many forms, chiefly non-numeric, as possible. It tends to focus on exploring, in as much detail as possible, smaller number of instances or examples which are seen as being interesting or illuminating, and aims to achieve 'depth' rather than 'breadth'."* (Blaxter et al. 2001)

Thus, open questions, and more generally, unscheduled standardized and non-standardized interviews lend themselves much more to qualitative research than scheduled interviews consisting mostly of ranking and multiple choice questions.

3.2.2 Questionnaire survey For questionnaire surveys, the principles outlined above still hold true, as they are basically a written form of a survey-type interview, aimed at

obtaining mainly quantitative data.

Nonetheless, questionnaires can – and often do – also contain qualitative questions.

3.3 Benchmarking

> "Benchmark test: A test of computer software or hardware that is generally run on a number of products to compare their performance." (McGraw-Hill Dictionary of scientific and technical terms).

> "Benchmarking is the continuous process of measuring products, services and practices against the toughest competitors or those companies recognised as industry leaders".
> *The Xerox Corporation*, as quoted in (Public Sector Benchmarking Service, 2005).

Ravden and Johnson (1989) devised a method for evaluating the usability of human-computer interfaces. They stated that *"if end-users find that the system actually interferes with, rather than enhances, their work, and if it causes them undue stress and frustration, then they may find it inefficient to use, and may actually cease to use it altogether"* (Ravden and Johnson 1989). Their evaluation method consists of a check-list, containing 11 sections. These are:

1 visual clarity,
2 consistency,
3 compatibility,
4 informative feedback,
5 explicitness,
6 appropriate functionality,
7 flexibility and control,
8 error prevention and correction and
9 user guidance and support
10 system usability problems
11 general questions on system usability

Sections 1 through 10 contain a check-list with multiple choice answers in the form of a scale containing 3 or 4 possibilities. Also, each question can further be commented on. Section 11 only contains open-ended questions.

Visual clarity is concerned with the display of information on the screen: whether the screen is cluttered, whether necessary information can be found easily and whether the user's attention is drawn to important information.

Consistency looks at the consistent use of items such as menus, colours, windows and abbreviations throughout the interface. Consistency helps the user orientating himself and lessens the effort necessary to learn operating the program.

Compatibility is concerned with whether the items used in the interface, e.g. colours or abbreviations, conform with existing norms and conventions. Here again, familiarity with the terms used in the interface puts less demand on the user, especially an inexperienced one.

Informative feedback deals with whether the information the user is given as a reaction to his usage of the interface is concise, clear and easy to understand. This section also looks at the amount of feedback given by the interface.

The section on **explicitness** is aimed at the structure of the system, i.e. whether the interface is clear and easy to understand, without the use of feedback. This section also deals with whether the structure is obvious to the user.

Appropriate functionality is concerned with whether the interface provides the functions, options and facilities the user requires in order to complete his tasks.

Flexibility and control looks at the flexibility to accommodate the needs and requirements of various users in different situations. These varying requirements, as stated by Ravden and Johnson, can for example be due to the experience of the user. They also state that too little flexibility as well as too much flexibility can cause difficulties.

The section on **error prevention and correction** deals with how the interface copes with preventing errors where possible and how it reacts to errors committed by the user. This section also looks at what facilities are provided to 'undo' inputs before processing.

User guidance and support is concerned with what on- and off-line help facilities are provided to the user, regardless of the user's experience and how they are structured.

The section on **system usability problems** looks at the interface in general and what type of problems (i.e. none/minor/major) the user has had with it while performing the evaluation.

The last section, **general questions on system usability**, is the only section without

multiple choice questions, offering only open-ended ones. This aims at providing space for more in-depth evaluation of the interface, as opposed to the small comments space in the other sections, which are reserved for details within each question.

Head (1999) took a slightly different approach in that she attributes more importance to the user base, i.e. the users' experience. Thus, her approach consists first of all of defining the user base, for which she established 10 questions. (see Table 3.2)

Table 3.2: Ten Questions (Head 1999)

To ask about the resource:	**To ask about the users and the setting:**
- What main tasks does the resource support? - What level of users is the resource aimed at? - Is the resource a tool for users with a certain expertise (e.g., professional graphic artists) or is it a general tool?	- What tasks are users counting on getting done with the resource? - What skill levels best describe the users (novice, intermediary, expert)? - Is there a match between the resource's purpose and the users' needs from a resource? - What training will be provided, in addition to the resource's training features or guides? - Will the resource be an end-user tool or will hands-on assistance be provided? - Does the resource run on platforms that users are familiar with? - What processing limitations about the user base can you identify?

Those questions, which should be asked during the first stages of the evaluation process, are centred around the experience of users with interfaces, as the usability, according to Head, depends heavily on the user group's proficiency.

She then suggests an analysis of the interface based on the categories of task support, usability and aesthetics.

The benchmark for task support, as put forth by Head (1999), are that the interface's functions match what the user wants to do, are *"easy to locate, comprehend and execute"*. Moreover, she established the following questions to evaluate task support:

- Are the functions easy to locate on the screen?

- Do the functions that receive top billing support users or the developer or the company?
- Are the functions easy to read?
- Are the names for functions easy to comprehend?
- Is decision making for users enhanced or impeded by the design?

These questions match to some extent the section on visual clarity and explicitness in the checklist by Ravden and Johnson (1989).

The part on usability is concerned with how easy the interface is to understand and work with. For the evaluation of this usability, Head (1999) devised the following questions:

- Are fundamental menu functions in the same location from screen to screen, such as print, help, quit and search?
- Are the same fundamental functions available from screen to screen?
- Is there a semantic clarity in the vocabulary that is used to describe functions, features, and icons from screen to screen?
- Overall, does the interface feel like a single interface or does interacting with the system feel like working with more than one system because of the design variability from screen to screen?

Thus, Head's category on usability contains parts of the consistency and functionality sections of the check-list by Ravden and Johnson (1989). Aesthetics, which deal *"with the use of color, icons, images, multimedia and the layout of elements"* (Head 1999), can be evaluated through the following questions established by her:

- How long does it take to process information appearing on the screen? Do graphical elements help or hinder processing?
- Are graphical elements - colour, layout, icons, fonts - used to prioritize key processing information?
- Are users overwhelmed by screens because of an overuse of colour?
- Do multimedia features - animation and sound - enhance or distract information processing?

The above questions on aesthetics are reflected by the sections on visual clarity, consistency and compatibility in the Ravden and Johnson (1989) checklist.

Ravden and Johnson (1989), however, have gone beyond establishing guidelines for evaluating the interface as such, as they also provide a guide for the construction of the

evaluation task itself. They suggested a two-pronged approach, consisting of a basic task analysis and development of evaluation tasks.

They proposed gathering and representing information about the intended use of the system under evaluation as well as on existing work on similar systems. From this information, a description is to be produced and then analysed. These tasks constitute the basic task analysis.

Developing the evaluation tasks consists of deciding on the number of tasks to be used for the evaluation and checking their representativeness and carrying out a pilot investigation in order to improve the evaluation tasks.

According to the UK's Public Sector Benchmarking Service, 2005, there are seven types of benchmarking:

1 Strategic Benchmarking, which looks at the long-term strategies of successful companies
2 Performance Benchmarking or Competitive Benchmarking, looking at the performance characteristics of successful products or services
3 Process Benchmarking is used for improving specific operations and processes
4 Functional Benchmarking or Generic Benchmarking looks at organizations in other areas, with the goal of innovation
5 Internal Benchmarking, which is done inside one company, e.g. among different business units
6 External Benchmarking serves as a comparison to the best competitors, i.e. looks for the 'best practice'
7 International Benchmarking, a form of external benchmarking done with companies based in different countries

The Public Sector Benchmarking Service, 2005 further differentiated those 7 types of benchmarking based on its focus and its extent, as shown in table 3.3

Table 3.3: Grouping of benchmarking types

Focus	**Extent**
Strategic Benchmarking	Internal Benchmarking
Performance Benchmarking	External Benchmarking
Process Benchmarking	International Benchmarking
Functional Benchmarking	

On their website, the Public Sector Benchmarking Service, 2005 stated that in general, benchmarking usually consists of:

- regular comparisons of performance aspects with best-of-practice
- identifying performance gaps
- development of new approaches for performance improvement
- following through with the implementation of improvements
- monitoring progress and reviewing benefits

3.4 Experiments

Yin (2003) included, in his book on case study research, a guide to help decide what form of research is best in specific cases:

Table 3.4: Relevant Situations for Different Research Strategies (Yin 2003)

Strategy	Form of Research Question	Requires Control of Behavioral Events?	Focuses on Contemporary Events
Experiment	how, why?	Yes	Yes
Survey	who, what, where how many, how much?	No	Yes
Archival analysis	who, what, where how many, how much?	No	Yes/No
History	how, why?	No	No
Case study	how, why?	No	Yes

Hence, if any form of control over certain variables in a research are required, experiments are the only option.

Experiments themselves can be divided into four categories wikiex06:

- Controlled experiments
- Natural experiments
- Observational studies
- Field experiments

3.4.1 Controlled experiments In the case of controlled experiments, the results from the experiment are compared to a control sample. In this case, all variables, apart from the one tested, are identical in both environments (Salford University 2006, Wikipedia.org 2006).

However, this class of experiments is also very useful when not all the conditions can be exactly controlled, in which case two probabilistically equivalent sample groups are created. Only then is the experiment carried out (Wikipedia.org 2006).

A typical use of controlled experiments can be found in the medical domain, where one group is given the actual medication, while a control group only receives a placebo.

3.4.2 Natural experiments This type of experiment is also called *quasi-experiment*, and are conducted when it is too difficult or impossible to conduct a controlled experiment. In such a case, the experimentation consists mainly of observing the variables under study, rather manipulating them. In the case of natural experiments, however, undetected variables can have a substantial influence on the outcome (Salford University 2006, Wikipedia.org 2006).

3.4.3 Observational studies This class is very similar to controlled experiments, but lacks the probabilistic equivalency between the groups. As a result, the results on their own are less reliable (Salford University 2006, Wikipedia.org 2006).

One example where observational studies are employed is, again, in the medical sector, where the *'control' group* is still given some medication, though different from the test group, rather than a placebo (Wikipedia.org 2006).

3.4.4 Field experiments This type of expirement is similar to a controlled experiment, only that it is carried out in the natural environment, as opposed to laboratory settings. While this results in a better applicability to the real world, it leads to the problem of having less control over all the variables that could – or can – have an influence on the outcome of the experiment (Salford University 2006).

Harrison and List (2004) defined three types of field experiments:

- artefactual field experiments (very similar to a conventional lab experiment, where there is a standard subject pool of students, abstract framing and and imposed set of rules, but with a non-standard subject pool – ie *'real'* people instead of students)
- framed field experiments (same as artefactual, but field context in commodity, task or information set that subjects can use)
- natural field experiments (as framed, but environment where subjects naturally undertake these tasks and subjects do not know they are in an experiment)

Framed field experiments should be undertaken *"in naturally occurring settings in which the factors that are at the heart of the theory are identifiable and arise endogenously, and then to impose the remaining controls needed to implement a clean experiment"* (Harrison and List 2004).

3.5 Justification for the methodologies used

3.5.1 Interviews/ Questionnaire survey In order to gain a good understanding of the problems simulationists face in terms of adaptability of the simulation packages they use, some form of direct information gathering was necessary. Semi-structured interviews allowed gathering some basic data, supplying the author with important indicators as to where there are potential problems in relation to adaptability of simulation projects. Also, the interviews helped outline the structure of the questionnaire survey, which was then used to obtain quantifiable data on various aspects relating to the adaptability of various simulation packages.

While the interviews provided some basic, valuable insight, it was deemed important to also gather information of a wider range of opinions on certain aspects of adaptability in simulation models. In order to achieve this, the questionnaire survey (see section 4.2) was conducted, to supply more specific, quantifiable data.

The evaluation of scaled multiple choice questions and ranking questions Keats (2000), ie where numerical data was obtained, which was rated 1 (hardest) through 4 (easiest), was done by calculating a weighted average to get an overall indication as to how a certain task was perceived by the simulation community. Simultaneously, these answers were also grouped $1 + 2$ (*very difficult* and *somewhat difficult*) and $3 + 4$ (*somewhat easy* and *very easy*) to get an overview of the overall inclination in terms of difficulty, as the numerical value alone, obtained through the weighted average, was considered insufficient.

In those cases where no numerical data could be obtained, ie where the answers were a simple *"yes"* or *"no"*, only the percentage of a certain answer per total of respondents was taken.

3.5.2 Benchmarking A comparison of some of the most widely used simulators in the form of benchmarking offered the best opportunity to gain first-hand knowledge of the problems simulationists in various areas face with those packages. Furthermore, a

benchmarking study was considered as the most valuable method to establish a ranking in terms of the packages' adaptability. This is because a check-list, against which each package is evaluated on the same, basic tasks, offers the most objective measurement and easiest comparison.

While the check-lists by Ravden and Johnson (1989) and Head (1999) are very adequate for evaluating the interface and the overall ease of use of various packages, they do not give any indication as to the intrinsic adaptability of the software packages. It was therefore deemed necessary to develop a alternative check-list for the evaluation of a model's intrinsic adaptability and to conduct an additional benchmarking study. This was based on the type *Performance Benchmarking* as identified by PSBS05. For this, a specific set of modifications to identical simulation models was performed in the two packages that were evaluated.

3.5.3 Experiments In order to analyse the best possible approach to providing more adaptability in a simulation model, an in-depth study of a simulation model was necessary. No other approach would have allowed comparing the effects on adaptability of creating a model for single use and modifying it versus the creation of a model with the idea of later modification in mind.

As a result, case studies, where there is little influence on the control variables, could not be used.

In order to allow a wide range of possible changes to be used in combination, so as to create conditions similar to those in industry, thereby giving the study a representative value, only some form of experiment could be used.

To overcome the problem that students are not representative of the whole simulation user group, conventional laboratory experiments were not applicable. Also, to see how the guidelines developed in chapter 6 on page 94 work when applied to the real world, it was necessary to test them in a natural setting. Thus, only a framed or natural field experiment could be used. It was, however, impossible to test the guidelines without the test subjects knowing about this being an experiment, and thus, by default, the *framed field experiment* had to be used.

However, some elements of a controlled experiment were also used, as the test group was going to be compared to a control group.

Chapter 4

Interviews and Questionnaire Survey

4.1 Interviews

Based on the findings pertaining to the correct layout and structure of interviews and the questions (see § 3.2 on page 41), a semi-structured interview was planned to serve as preparatory stage for a much wider simulation user survey. The questions types and sequences identified in section 3.2 on page 41 were of importance to the planning and conducting of interviews as well as to the planning of the survey.

4.1.1 Reasons for interview questions

Experience: with an average duration of 6 months per simulation project, a person having done 30 would have been practicing simulation for 15 years. However, most simulation modellers tend to move on to other positions after approximately 5 years of practicing simulation. For this reason, the categories for the simulationists' experience in terms of simulation projects were split into:

- Less than 4 models built
- Between 4 and 7 models built
- Between 7 and 10 models built
- Between 10 and 20 models built
- Between 20 and 30 models built
- More than 30 models built

Simulation program/package used: the program or package used can have an effect on the overall duration of the project, in terms of the time needed for coding or for implementing changes to the model, depending on its adaptability.

Project length: to get a feel for the minimum, average and maximum lengths of projects and to find out what causes very short or very long project times.

Most suitable process description (see fig. 2.1 on page 9, fig. 2.2 on page 12, fig. 2.3 on page 14 and fig. 2.5 on page 16): so that the modeller can pick a description that fits his/her approach best; a suitable description of their process is important for next questions.

Type of changes:

To which phase/step of cycle: Identify the area most prone to the need to change on the 'most suitable process description', in order to establish an importance rating for those changes.

What is affected by the change: Does the change involve the objectives, the logic, number of machines or products/entities etc.

Which is easiest/most difficult to implement: To establish importance rating for difficulty

Recurrence of change: to establish which change is most likely to occur, not just which is most difficult to implement.

What/who influences those changes: Is it new circumstances, e.g. better understanding of the project, through additional feedback by problem owner's team? Is it a change of objective altogether?

Importance of graphics/animation: If animation is not important, e.g. not used, then there will be no request to change it. It might be used only for validation, and/or for presentation of results. This is also important for the next question.

The simulation:

Is it for further use by the modeller / to be passed on to another modeller for further use / to be passed on to a group of people for further use: Depending on whether further use will be by other people, and if, who they will be and how many, there can be a need to adapt the model in terms of presentation, to make it more transparent to work with. For example, if the simulation model was created only for use by the modeller himself, then the structure does not need to be clear to other people. If, on the other hand, the model will be used on the shop floor level, then a high degree of usability, such as obvious structuring, is necessary.

Due to the semi-structured properties of the interviews, a multitude of question sequences was used. The question on the simulation user's experience, for example, was a simple structure, as an answer was given and then the next theme was started. On other occasions, eg the maximum project duration, a chain structure, or if a complex answer requiring additional probing was given, a branching or sequential structure was used. Also, as the basic, initial questions were grouped together in themes, the constellated structure also applies. The complex branching structure, however, was only rarely used.

4.1.2 Interview results In order to gather some basic data for the design of the questionnaire survey, 3 experienced simulation practitioners, working with different simulation software packages, were interviewed.

During the interview, notes were taken of the answers as well as of the probing questions that arose from various answers.

Fragmentary example of a transcript:

experience: 4 yrs, approx. 10 projects.
Package used: AutoMod
shortest project length: 2 ... 3 weeks, reuse of existing simulation models
Probing question: How were they reused?:
adapted to look similar to the proposed system
Probing question: Why were they made to look like the proposed system, and not represent it?
This is only used for the first stage, for sales.

Overall, it emerged that the model by Banks and Carson (see § 2.3.3 and figure 2.3 on page 14) was favoured by all interviewees over the models by Law and Kelton (§ 2.3.1 and figure 2.1 on page 9), Balci (§ 2.3.2 and figure 2.2 on page 12) and Benjamin et al. (§ 2.3.5 and figure 2.5 on page 16).

All three interviewees expressed interest for animation mostly as a tool for validation and to give people outside the specific domain (e.g. general managers) an idea. For the personnel directly involved in the simulation project, the statistical output proved more important. They also complained about shortcomings of their respective simulation software packages in the adaptability of existing models. The interviewees further expressed that the limited adaptability of modelling software and/or the lack of specific templates

substantially increased the model building time.

Also, it emanated that having access to the code was a necessity, even in software with graphical model building properties, such as Arena™.

4.2 Questionnaire Survey

It was deemed important to conduct the benchmark study on the simulation software packages most widely used at present. In order to gather up-to-date information on the simulation software packages used by academics and industry, a questionnaire survey was conducted during the 2003 Winter Simulation Conference, December 7 through 10 at the Fairmont Hotel in New Orleans, via online-questionnaires over the internet and during the International Conference on Manufacturing Research (ICMR2004) in Sheffield. The data, which was obtained between Dec. 2003 and Sept. 2004, is based on a sample size of 92 responses, with the exception of the software ranking (see table 4.1 on page 67). Out of those 92, only 87 were valid and thus used subsequently. That is, partially filled-in forms, or ones that contained contradicting results, were discarded. The results for the ranking were based on a sample size of 46, which had been collected between Dec. 2003 and May 2004. Data collected after this period were not taken into account for the ranking, as software-specific user groups were contacted for the questionnaire, which would have skewed the spread of specific packages towards those used by the few groups contacted.

The aim of this questionnaire (Pohl 2003) was to investigate the spread of various software packages and where the biggest problems in implementing changes to the models occur. Those *"biggest problems"* were looked at in terms of frequency of recurrence and severity. The frequency of recurrence was simply the number of ticks in any category or the ratio of ticks in the categories *never* and *sometimes*, and *frequently* or *always*. The severity was rated as the percentage of ticks in the categories *somewhat difficult* and *very difficult* as opposed to the amount of ticks in the categories *somewhat easy* and *very easy*.

The majority of the questions in the survey were taken from the preliminary interviews, with some left out for irrelevance to statistical analysis. Other questions were introduced, or refined from questions asked in the interviews. This was necessary to have clear categories for respondents to tick.

The questionnaire was divided into 2 main parts:

- User background

- Main block of questions

However, the questionnaire also contained additional space so that respondents could express particular problems, limitations or advantages they see in their software.

In terms of question sequence, the simple structure, as well as the constellated structure, where used.

4.2.1 Questions — User background In order to gather some basic distinctions among the simulation software users, two distinction methods were used: User type and model-building experience. The user categories were, as in the case of Hlupic (2000), *industrial* and *academic* users.

To differentiate the users by means of model building experience, the following categories were created:

- Less than 4 models built
- Between 4 and 7 models built
- Between 7 and 10 models built
- Between 10 and 20 models built
- Between 20 and 30 models built
- More than 30 models built

Another question relating to the user background was the main simulation software product(s) used.

4.2.2 Questions — Main bloc The next series of questions focused on the difficulties to add, modify or delete various logic elements (i.e. entities, resources and handling devices) with a range of $4 = easiest$ to $1 = hardest$. By attributing a higher number to the easier task, a higher value shows that is was easier to make one change than another, thus representing a higher adaptability.

This bloc also included questions pertaining to the ease of changing input data (modifying the distribution as well as changing from internal to externally stored data), the complexity related to changing the amount of detail of an existing simulation model and the frequency of modifying an existing model to reflect a change in the underlying system as well as the main reasons for reconfiguring a model.

Another part of this main question bloc also focused on the availability – and use – of generic model components, e.g. templates.

Further questions included the difficulty to change an animation, the availability of undo/redo options and the frequency with which respondents tended to modify the simulation code itself.

4.2.3 Results — User background Approximately 40% of respondents came from academic background and the rest were industrial simulation software users. In this instance, multiple responses were possible, i.e. respondents who worked with simulation in an academic area as well as in an industrial setting, were counted as belonging to both groups.

The experience of simulation software users varied greatly, as much among the individual groups (academic/industrial) as overall. See results in figure 4.1.

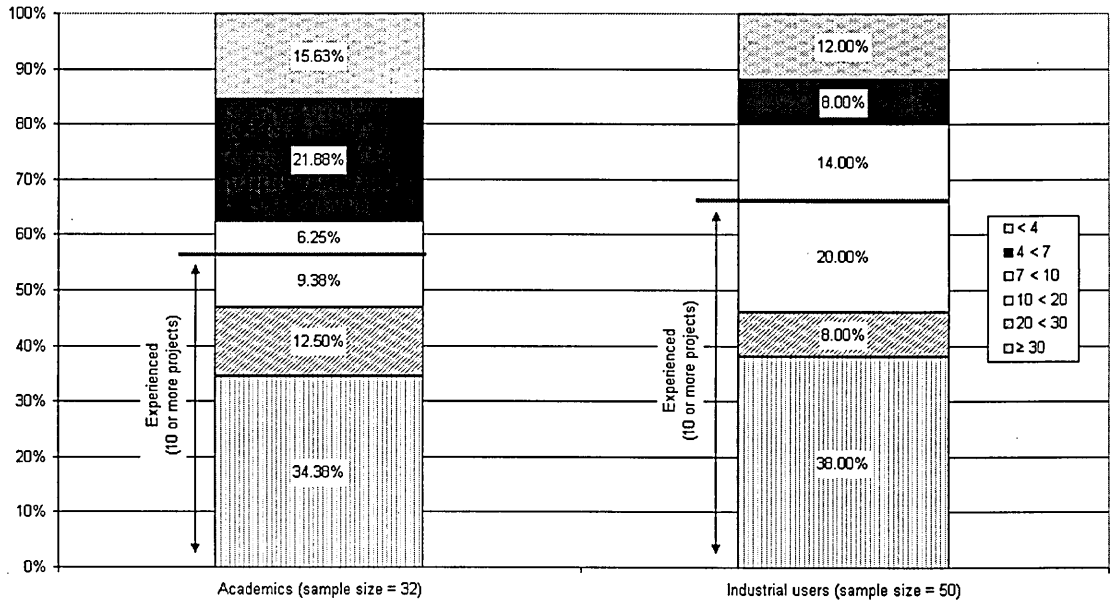

Figure 4.1: Overall model building experience

The software ranking (see table 4.1 on the following page) was based on the replies to the main software package used. As two answer slots were provided, the first entry was taken as 'first choice' software and the second, if filled in, as 'second choice'. If, in the paper version, no clear distinction was made between first and second choice by the respondent, the answer was altogether ignored for the ranking shown above.

Table 4.1: Software ranking

First Choice	%	Second Choice	%	Total	%
Arena	39.02	Arena	11.11	Arena	30.51
Extend	7.32	Extend	11.11	Extend	8.47
Authorware	7.32	AutoMod	11.11	Authorware	6.78
Simul8	4.88	Authorware	5.56	Simul8	6.78
FlexSim	4.88	Simul8	11.11	AutoMod	5.08
Quest	4.88	MatLab/	5.56	FlexSim	3.39
OneSAF	4.88	Simulink		Quest	3.39
Witness	4.88	Devs JAVA	5.56	OneSAF	3.39
AutoMod	2.44	HLA	5.56	MatLab/	3.39
MatLab/	2.44	MicroSaint	5.56	Simulink	
Simulink		ProModel	5.56	Witness	3.39
GPSS/H	2.44	TopQ	5.56	ProModel	3.39
MONARC	2.44	Proof	5.56	Others	22.03
VR-Forces	2.44	Netlogo	5.56		
RePast	2.44	JSAF	5.56		
Service Model	2.44				
SAS	2.44				
ProModel	2.44				

Others (at 1.69 % each):

- GPSS/H
- MONARC
- VR-Forces
- RePast
- Service Model
- SAS
- Devs JAVA
- HLA (DMSO RTI1.3)
- MicroSaint
- TopQ
- Proof
- Netlogo
- JSAF

Statistics indicating the number of times various simulation packages have been mentioned in the Proceedings of the 2005 Winter Simulation Conference, as shown in figure 4.2 on the next page, confirm the overall high ranking and widespread use of Arena simulation software.

Comparison (by Product) from the 2005 *Proceedings of the Winter Simulation Conference*

Frequency of mention, by product:

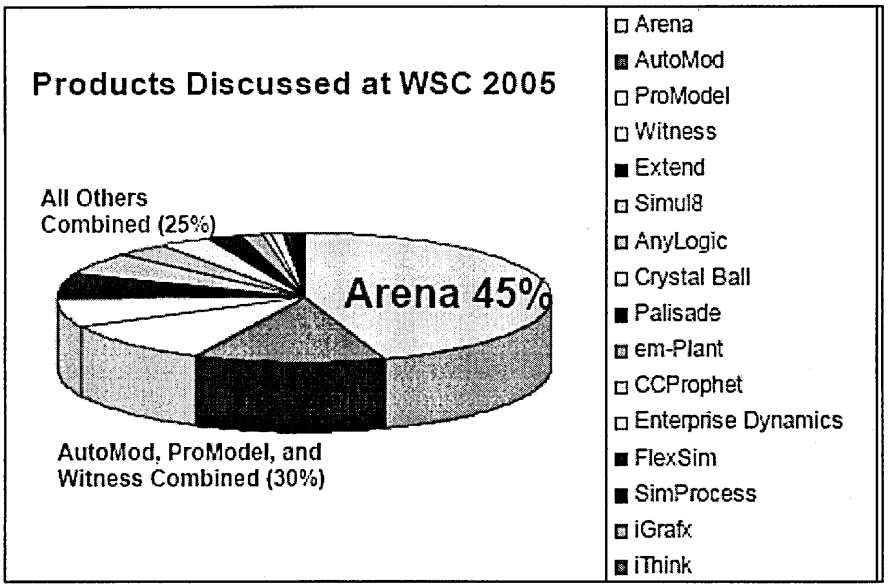

Figure 4.2: Software ranking (Rockwell Software 2006)

4.2.4 Results — Main bloc The statistics listed below are based on the total of responses, ie on all 87 valid samples, irrespective of experience and user type.

The question about adding entities, resources or handling devices is based on the basic task of having to create either of these elements in order to build or modify a simulation model. The same also applies to modifications and deletions of elements. (see also table 4.2 on the following page)

The values were arrived at as shown in equation 4.2.1 for all tables except table 4.3 on page 70.

$$v_1 = \alpha \times \frac{100}{\beta} \tag{4.2.1}$$

v_1 is the percentage shown in the table, α the number of counts in one category (eg *very easy* for *adding elements*) and β the total number of respondents.

For instance, 54 out of 87 people considered adding entities *very easy*. Thus, $\alpha = 54$ and $\beta = 87$ and therefore $v_1 = 54 \times \frac{100}{87} \Longleftrightarrow v_1 = 62.07\%$

The percentages in table 4.3 were calculated as described in equation 4.2.2.

$$v_2 = \frac{\sum \gamma + \sum \delta}{\sum \sigma} \times 100 \tag{4.2.2}$$

In this case, v_2 is the percentage value shown in the table, $\sum \gamma$ the number of *very easy*

Table 4.2: Difficulty in modifying elements (v_1)

Change	Difficulty	entities	resources	handling devices
Add	4 (very easy)	62.07 %	55.17 %	28.74 %
	3 (somewhat easy)	27.59 %	24.14 %	27.59 %
	2 (somewhat difficult)	4.60 %	12.64 %	29.89 %
	1 (very difficult)	5.75 %	8.05 %	13.79 %
Modify	4 (very easy)	48.28 %	47.13 %	22.99 %
	3 (somewhat easy)	40.23 %	37.93 %	36.78 %
	2 (somewhat difficult)	5.75 %	10.34 %	31.03 %
	1 (very difficult)	5.75 %	4.60 %	9.20 %
Delete	4 (very easy)	68.97 %	62.50 %	45.98 %
	3 (somewhat easy)	16.09 %	17.05 %	28.74 %
	2 (somewhat difficult)	4.61 %	13.64 %	19.54 %
	1 (very difficult)	10.34 %	6.82 %	5.75 %

instances across a range of categories (eg *add entities+add resources+add handling devices*). $\sum \delta$ is the number of *somewhat easy* instances across the same range of categories, and $\sum \sigma$ the total of all instances (ie *very easy, somewhat easy, somewhat difficult* and *very difficult* combined) for that range of categories.

For the category 'add elements', v_2 is:

$$v_{2_{(add\ elements)}} = \frac{\sum \gamma_{(add\ elements)} + \sum \delta_{(add\ elements)}}{\sum \sigma_{(add\ elements)}} \times 100$$

$$\iff \sum \gamma_{(add\ elements)} = add\ entity_{(very\ easy)} + add\ resource_{(very\ easy)} + add\ handling\ device_{(very\ easy)}$$

$$\Rightarrow \sum \gamma_{(add\ elements)} = 54 + 48 + 25$$

$$\Rightarrow \sum \gamma_{(add\ elements)} = 127$$

$$\iff \sum \delta_{(add\ elements)} = add\ entity_{(somewhat\ easy)} + add\ resource_{(somewhat\ easy)} + add\ handling\ device_{(somewhat\ easy)}$$

$$\Rightarrow \sum \delta_{(add\ elements)} = 24 + 21 + 24$$

$$\Rightarrow \sum \delta_{(add\ elements)} = 69$$

$$\iff \sum \sigma_{(add\ elements)} = add\ entity_{(very\ easy)} + add\ resource_{(very\ easy)} + add\ handling\ device_{(very\ easy)} add\ entity_{(somewhat\ easy)} + add\ resource_{(somewhat\ easy)} + add\ handling\ device_{(somewhat\ easy)} + add\ entity_{(somewhat\ difficult)} + add\ resource_{somewhat\ difficult} + add\ handling\ device_{(somewhat\ difficult)} + add\ entity_{(very\ difficult)} + add\ resource_{(very\ difficult)} + add\ handling\ device_{(very\ difficult)}$$

$$\Rightarrow \sum \sigma_{(add\ elements)} = 54 + 48 + 25 + 24 + 21 + 24 + 4 + 11 + 26 + 5 + 7 + 12$$
$$\Rightarrow \sum \sigma_{(add\ elements)} = 261$$

$$\Rightarrow v_{2_{(add\ elements)}} = \frac{127+69}{261} \times 100$$
$$\Rightarrow v_{2_{(add\ elements)}} = 75.0957854\%$$
$$\Longrightarrow v_{2_{(add\ elements)}} \approx 75.10\%$$

The single figure digit v_3 was calculated as indicated in equation 4.2.3.

$$v_3 = \frac{v_2 \times 3}{100} + 1 \tag{4.2.3}$$

As the scale in the questionnaire was 1 to 4, ie with a difference of 3 units, this had to be taken into account for the calculation by adjusting the scale to $0 - 3$ instead of $1 - 4$. 1 was added to give a figure in the previously used range of $1 - 4$.

For the category 'add elements', v_3 is therefore:

$$v_{3_{(add\ elements)}} = \frac{v_{2_{(add\ elements)}} \times 3}{100} + 1$$
$$\Rightarrow v_{3_{(add\ elements)}} = \frac{75.0957854 \times 3}{100} + 1$$
$$\Rightarrow v_{3_{(add\ elements)}} = 3.25287356$$
$$\Longrightarrow v_{3_{(add\ elements)}} \approx 3.25$$

Table 4.3 shows the weighted difficulty ratings for various tasks and elements. The weighting is based on the percentages from table 4.2 on the preceding page, for the groups *very easy* and *somewhat easy* combined. The *weighted rating* gives a figure an the range 1 to 4, as described above (see equation 4.2.3).

Table 4.3: Weighted difficulty ratings

task/element	**easy (v_2)**	**weighted rating (v_3)**
Add elements	75.10%	3.25
Modify elements	77.78%	3.33
Delete elements	79.77%	3.39
Entities	87.74%	3.63
Resources	81.30%	3.43
Handling devices	63.60%	2.91

Changing data form refers to switching from internal (e.g. distribution) to external data (e.g. data files), and/or from stochastic to empirical data.

Table 4.6 on the next page refers to the frequency of modifying a working, validated model.

Table 4.4: Difficulty in modifying a distribution and changing the data form (v_1)

Difficulty	distribution	data form
4 (very easy)	58.14 %	27.06 %
3 (somewhat easy)	22.09 %	31.76 %
2 (somewhat difficult)	12.79 %	29.41 %
1 (very difficult)	6.98 %	11.76 %

Table 4.5: Difficulty in adding and removing model detail (v_1)

Difficulty	adding model detail	removing model detail
4 (very easy)	20.69 %	23.26 %
3 (somewhat easy)	40.23 %	43.02 %
2 (somewhat difficult)	25.29 %	27.91 %
1 (very difficult)	13.79 %	5.81 %

Table 4.6: Frequency of modifying model logic and simulation code

Frequency	model logic	simulation code
never	3.41 %	17.24 %
sometimes	39.77 %	48.28 %
frequently	40.91 %	25.29 %
always	15.91 %	9.20 %

Table 4.7: Availability of 'undo' and 'redo' options and of templates

Availability	undo option	redo option	templates
yes	62.24 %	43.62 %	79.78 %
no	37.76 %	56.38 %	20.22 %

Table 4.8: Usage of available templates

Frequency	Percent
never	16.44
sometimes	28.77
frequently	38.36
always	16.44

Table 4.9: Main reason for reconfiguring the model logic

Reasons	Percent
Major change to underlying system	20.83
Change to amount of detail required	21.88
Change to simulation objectives	42.71
Other	14.58

Table 4.10: Difficulty in modifying animations (v_1)

Difficulty	Percent
4 (very easy)	26.14
3 (somewhat easy)	45.45
2 (somewhat difficult)	23.86
1 (very difficult)	4.55

4.3 Key Findings

4.3.1 Interviews From the first stage of interviews, it emanated that the steps in a simulation study as proposed by Banks and Carson (1984) was favoured over the approach suggested by Law and Kelton (2000).

The interviews also showed that, when for their own use, the interviewees give little to no attention to the animation, and rely solely/mainly on the statistical output. As a result, animation is only considered important when the finished model is to be presented to people outside the specific domain in which the project is conducted.

The third key finding from conducting the interviews was that all interviewees emphasized the need to access the simulation code, i.e. the ability to modify the basic simulation code.

4.3.2 Questionnaire Survey The questionnaire showed that, with a third of all simulation users working with Arena, this package is the most widely used one, even though it has a somewhat bigger following among academics than industrial users. Also, it emanates that more than 50% of all simulation users work with commercial off-the-shelf simulation packages.

The survey further showed that deleting any type of element is the easiest task, while modifying is the most difficult one. Also, tasks relating to entities (elements that flow through the simulation) are the easiest to modify, while tasks relating to handling devices pose the biggest problems.

Another interesting finding is that more than a third of simulation users change the layout frequently, and just over 3% do not modify the logic of a valid model at all. In 43% of cases, these changes are made necessary by a change to the simulation objectives over the course of the study and in 21% it is due to a major change in the underlying system. For 22% of the interviewees, such changes to the model logic have to be made to account for requirements in terms of the detail required, while the other interviewees gave a multitude of reasons for modifying the logic, which could not be grouped in any significant way.

Chapter 5

Benchmarking

The interviews (see chapter 4) served to gain insight into how experienced simulation practitioners go about creating their models, and to gather some ideas and data for the design of the user survey, whose results are also stated in chapter 4. The questionnaire was used to evaluate the users' perspectives on various tasks associated with modifying simulation models, such as which tasks were easiest/most complicated, which tasks were most frequent, and which software groups and packages were most frequent and easiest to use. All these elements had not been previously investigated to the extent necessary for this thesis.

The user survey revealed, among others, that COTS packages were the most frequent type of simulation software used. The next stage is therefore to evaluate and compare the specific adaptability of two of the most widely used discrete-event simulators. This is done first of all by analysing the packages according to various definitions of adaptability and literature, then by implementing a specific list of changes in each package and calculating the adaptability thereof. Finally, a controlled experiment is conducted on one of the packages to test the influence of one specific element, subgroups, on the overall – and perceived – adaptability of this one package.

5.1 Definitions

In section 2.4, different types of adaptability and their definitions were looked at. In this thesis, the term 'adaptability' refers more specifically to the ability to quickly modify an existing, valid simulation model to fit changing, or changed, situations. That is, it includes as much a change in the underlying system, where machines or operators (resources) are added, removed, or modified, as particular data, such as statistical distributions, or

changing from stochastic to empirical data.

As the check-list developed by Ravden and Johnson (1989) was inadequate (see section 3.5.2), as it focussed only on the evaluation of human-computer interfaces, it was not used for the benchmarking. However, it was very useful in terms of aiding in the development of an alternative check-list for the evaluation of a simulation model's adaptability.

This author suggests the following elements for such an evaluation:

1 **Data location adaptability**: The placement of input data within the model or as a separate part, i.e. separated from the model logic.
2 **Data type adaptability**: The ease with which data can be replaced. For example, whether strings are admissible as input data.
3 **Logic adaptability**: How easily the logic of the simulation model can be changed to adapt it to changes in the underlying system or changes to the main aims and goals of the simulation project. This includes the ability to change the number and type of entities, resources etc. included in the model.
4 **Animation adaptability**: The ease with which the animation can be modified to adapt it to a change in the model logic or the underlying system.
5 **Interface adaptability**: The ease of creating various interfaces to suit simulation software users who will take over the model for further experimentation.

5.2 Changes influencing the simulation model

A simulation is, as stated by Banks and Carson (1984), the *"imitation of the operation of a real-world process or system over time"*. Thus, any change to the real system should or can, depending on the detail of the model, be reflected in the simulation model. However, there are also changes which affect the model only, and not the underlying system. Therefore, it is possible to distinguish between system dependent changes and simulation model dependent changes. The former refers to modifications of the simulation model which are due to a change in the underlying real-world system. Simulation model dependent changes are changes to the simulation model which are not related to the underlying real-world system or modifications thereof.

These two groups can be further detailed as follows:

System dependent changes:

- product dependent changes
  - addition of a new product
  - elimination of an existing product
  - product modification
  - product substitution
- production process dependent changes
  - order of machines in the production process
  - number of machines
    - substitution
    - addition of machines
    - removal of machines

The substitution of machines can be further broken down into:

- single substitution *(e.g. a new machine replacing an old one, but with the same capabilities)*
- multiple substitution
  - one for many *(e.g. one new machine replacing X old ones)*
  - many for one *(e.g. X new machines replacing a single old one)*

Within the system dependent changes, one group is related to the product range of a given company. In the manufacturing sector, such changes are the introduction of a new product, the elimination of an existing product from its range, the modification of an existing product and the substitution of an existing product.

In car manufacturing, the introduction of a new product would mean the addition of a new model, for example, the introduction of a coupé on the basis of an existing 4-door limousine.

Product withdrawal would result in a reduction of the models offered. In the case of the car manufacturer, this could be the withdrawal of the coupé from its product range, for example because of poor sales.

Product modification, on the other hand, only changes some part(s) of an existing product, without significant change to its use. For example, the coupé being equipped with larger break discs would not alter its use, yet it still constitutes a change to the existing model.

Product substitution, finally, consists of withdrawing one product and replacing it with another. For the car manufacturer, this could, for example, be withdrawing the coupé from the market and replacing it with a 4-wheel-driven pick-up truck.

Production process dependent changes will not affect the products at the end of the manufacturing process, but they way they are produced. This may include changing the order in which the products go through the shop floor in order to be transformed into the final product. For example, product *x*, which was produced on machines A, B, C and D in this order will be produced in the order A, C, D and B. Such a change can be made necessary by logistics.

Another group of process dependent changes relates to the number of machines. If, for example, a company decides to integrate parts of the upstream manufacturing, i.e. sub-component manufacturing, it will have to add the machines for these sub-components. Also, if the workload increases beyond the capacity of a given machine, an identical one would be added to cope with the workload. If, on the other hand, the company decides to give out more work to subcontractors, it will enable the elimination of the machines necessary to produce that sub-component which will be bought from the sub-contractor.

Substitution is another issue, as one machine might be worn out and be replaced by an identical one. This would constitute a single substitution. Group substitution can go two ways: a new machining centre can replace a group of machines, thus freeing up space and, most of all, increasing production speed. On the other hand, it is also possible that the workload of one machine is split up on two machines, for example, instead of working all diameters ranging from 10 to 400 mm on one lathe, splitting the workload into 10 to 200 mm on one lather and diameter of more than 200mm on another lathe.

Simulation model dependent changes:

- objective
- logic/layout
- data
- end-user

Changes to the simulation model which are not caused by a change to the real-world system, but relate only to the simulation itself can be changes to the objectives. If, for instance, management is not interested in finding out the maximum output of its production

facility any more, but instead wants to calculate the difference in costs by reducing overtime work and introducing an additional shift, this constitutes a change of the objective of the simulation model.

Changes to the logic/layout can be the reorganisation of machines from simple lines into stations, or the association of sub processes into a sub model, instead of having them in the main model.

Changes to the data structure can entail eliminating a stochastic, probabilistic distribution and replacing it with an external file containing empirical data, or simply changing the type of distribution.

Implement any change is not as trivial as it seems. For instance, the addition of a product may result in having to modify substantial parts of the logic, if that product does not go through the same stations as the existing ones, or in a different order. If, for example, the modeller has created the logic with a series of "route" blocs (eg in Arena), then every single one of them might need to be changed to accomodate the inclusion of the new product. This could also lead to the inclusion of further decision blocs, or sequences being created.

Similarly, changing a handling device can result in a number of changes to the model logic as a whole. For instance, a handling device might require other resources to work, or affect buffers, which would then have to be modified accordingly, thus adding more steps to a seemingly trivial task.

The end-user also constitutes an important factor of change to a simulation model: For example, a model that was built to 'sell' a concept to management is adapted so that it can be used for daily evaluations on the shop floor. This could mean having to adapt the ease of use, so that a worker can manipulate it without needing the model builder, e.g. through the building of an interface so that the worker will not have to modify the simulation logic itself.

5.3 Choice of software

As outlined in §2.5.2, simulation models can be created in general purpose programming languages, special purpose simulation languages or by using high-level simulation packages. Of these three possibilities, the creation of models in high-level simulation packages offers the most user-friendly approach, as the least programming knowledge is necessary.

Furthermore, as pointed out by Kelton et al. (2002) general purpose programming languages offer the greatest amount of flexibility, with the same applying to special purpose simulation languages. In these two groups of languages, the amount of flexibility therefore depends exclusively on the programming prowess of the model builder. It was thus decided to evaluate the adaptability offered by high-level simulation packages only.

Hlupic (2000) identified Simul8 (44.4 %), Witness™ (38.8 %) and Siman/Cinema (33.3 %) as the most widely used simulation packages in the UK (see also § 2.7.1 on page 34). The survey conducted by this author (see § 4.2 on page 64) identified Arena™ (29.27 %), Extend™ (12.20 %) and AutoMod™ and Simul8™ (both 7.32 %) as the leading simulators worldwide. As the majority of simulation software with widest use is only two-dimensional, the packages Arena™ and Simul8™ were chosen for the evaluation.

5.4 Conducting the benchmarking

The benchmarking exercise was conducted in four stages; the first consisted of comparing the packages by means of their flexibilities as outlined in sections 2.4 and 5.1.

In a second stage, the findings obtained by Swain (2003) were listed (see also section 2.7.1).

For the third stage, specific tasks were performed, as outlined below.

In a final stage, the influence of sequencing on the adaptability was looked at.

A number of possible changes can affect a simulation model. Therefore, in order to evaluate the adaptability of different simulation packages, a basic model was built in each of them and then modified according to the possible modifications listed in section 5.2.

The basic list of changes to be implemented was broken down as follows:

- addition of a new product requiring no new resource
- elimination of an existing product
- modification of an existing product (e.g. some attributes of the "entity")
- addition of a new resource
- elimination of an existing resource
- modification of an existing resource (e.g. some attributes of the "resource")
- elimination of an existing product and a resource used only for this one product

In addition to these basic tasks, the following items also have to be considered:

- sequencing operations
- unavailability
  - schedules
  - breakdowns and repair times
- pooling of resources (sets)
- entity transfers
  - along guided paths (e.g. AGVs)
  - along free paths (e.g. trolleys)
  - on conveyors
- introduction of variables
- batching
  - temporary
  - final

Changing the number of resources, entities or handling devices is the first step, but it is just as important to evaluate the ability of the software to cope with attributes of those basic elements. For resources and handling devices, scheduling and the ease to include and modify breakdown and repair times, also become important.

The pooling of resources is concerned with allowing entities to go to any of a number of machines that can do the same task, e.g. within one station, using various constraints. Such constraints can be queuing rules such as *highest / lowest number busy* or highest / lowest remaining capacity in addition to the more general queuing rules of FIFO (First In First Out), LIFO (Last In First Out) etc.

Entity transfers in themselves, as well as their representation in animation, also need to be looked at. For example, an AGV (Automated Guided Vehicle) can only transfer entities along specified paths, thus leading to the possibility of a pile up if there is an obstacle on that path (e.g. another AGV in front of it), while transporters travelling along free paths can go around obstacles.

Each change was implemented in the basic simulation model, consisting of 5 stations. (see figure 5.1 on the following page) This eliminated the risk of cross-influencing, i.e. potential influences from other changes. The evaluation of complexity of implementation consisted of three main steps:

1 Counting the number of parts (i.e. logic elements) affected by the modification
2 Counting the number of steps necessary to implement the change in each affected part of the simulation model

3 Counting the number of steps necessary to reflect this change in the animation where applicable.

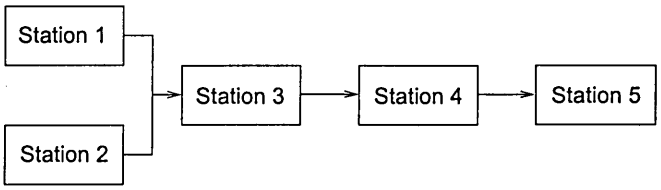

Figure 5.1: Basic model for benchmarking

For this initial model, there is one resource in each station, with the resource in Station 3 combining the products. Therefore, after Station 3, there is only one product going through the system. The modifications to this initial model include, among others, having more than one resource in a station, and more then 1 product entering Station 4.

The effort needed to adapt the models was measured using an equation based on one proposed by Herrmann et al. (2000). See also section 2.6 and equation 5.4.1.

However, the equation 5.4.1 devised by Herrmann et al. (2000) would result in a low number for high adaptability and vice versa, the author modified the original equation, as shown in equations 5.4.2 to 5.4.4 so that a high adaptability results in a high adaptability quotient, whereas a low adaptability (or high inadaptability I) will result in a high $I_{(M_0,M_1)}$ and low $A_{(M_0,M_1)}$.

$$A_{(M_0,M_1)} = \frac{E_{(M_1-M_0)}}{E_{(M_1)}} \tag{5.4.1}$$

The first step was to rename the first equation as follows:

$$I_{(M_0,M_1)} = \frac{E_{(M_1-M_0)}}{E_{(M_1)}} \tag{5.4.2}$$

From this follows:

$$A_{(M_0,M_1)} = 1 - I_{(M_0,M_1)} \tag{5.4.3}$$

and

$$A_{(M_0,M_1)} = 1 - \frac{E_{(M_1-M_0)}}{E_{(M_1)}} \tag{5.4.4}$$

This new adaptability formula, as shown in equation 5.4.4, allows for three distinct

scenarios:

$$A_{(Model)} = \begin{cases} > 0 & \Longrightarrow \ Change \\ = 0 & \Longrightarrow \ Decide \\ < 0 & \Longrightarrow \ Create \end{cases}$$

If the adaptability value is greater than zero, the model should be modified; if it is equal to zero, it is up to the model builder to decide whether to build a new model or to modify it. In the case of a negative adaptability value, it is better to create the model from scratch, rather than modifying an existing model.

In order to eliminate any variations caused by a differing amount of experience from one evaluation to the next, dry runs of each change were made. This way, the shortest and best way to implement a modification was established, thereby also eliminating additional steps stemming from errors.

Sequencing operations become necessary, when different products (entities) flow through the system, with a different order to the stations visited by the entities, or different entities using a varying number of stations.

5.5 Organizational layout

Most COTS-packages provide some form of grouping of modules into higher-level elements (eg sub-models and customized templates in Arena), which are aimed at simplifying the structure of the model.

While a model does appear simplified when customized templates or sub-models are used, there was no empirical data available relating to whether such an organization makes the modification of the model easier or not.

It was therefore deemed necessary to conduct a benchmarking for that. As the effect of such tiered grouping can be expected to be similar for all COTS-packages, only one, Arena, was used in this case. The choice was made for Arena due to better access to experienced modellers using this package, rather than Simul8.

Also, as the creation and modification of customized templates requires additional software modules, this benchmarking was conducted using only sub-models.

An initial model, very similar to the one used for the above mentioned benchmarking exercises, was used. Furthermore, two versions of that same model were built; one with,

the other without the use of sub-models. One group of experienced Arena users was then asked to modify the model containing sub-models, while a control group was asked to perform the same set of modifications to the model without sub-models.

5.6 Results overview

As mentioned in section 5.3, the COTS packages used for the evaluation were Arena™, version 7.0 (Rockwell Software) and Simul8™, version 6 (Simul8 Corporation). In neither case was the most recent version used, but the focus of this benchmarking was on the intrinsic capabilities rather than extended features such as VBA. As there have been only negligible changes to the intrinsic capabilities, the older versions were deemed sufficient for the purpose.

5.6.1 Package adaptability

1 Data location adaptability: Both, Arena and Simul8, allow for data to be placed inside the simulation model or in an external database/spreadsheet. In both cases, the external data is linked to the model via VBA code. In the case of Simul8, however, external databases can only be linked to the simulation model in the professional version, while spreadsheets (specifically Microsoft's Excel) can be linked in any version.

2 Data type adaptability: Arena does not permit strings as input data, while Simul8 does, to some extent, allow them. (e.g. for naming schedules or resources)

3 Logic adaptability: See section 5.6.3

4 Animation adaptability: See section 5.6.3

5 Interface adaptability: Both packages allow the creation of interfaces (e.g. in Excel) via VBA or other programming languages, so that the end-user has little or no need to manipulate the simulation model itself.

5.6.2 Results from the sixth biennial survey The results listed in table 5.1 on the next page are taken from Swain (2003) in their entirety. They show that, for the properties listed here, there is no difference between the two packages. Swain's survey, however, also includes some more details, which give Arena a very slight advantage over Simul8.

Table 5.1: Results from the 6th biennial survey of discrete-event simulation software tools (Swain 2003)

	Arena	Simul8
Graphical model construction	Y	Y
Access to programming modules	Y	Y
Runtime debugging	Y	Y
Input distribution fitting	Y	Y
Output analysis support	Y	Y
Batch run option	Y	Y
Optimization	Y	Y
Code reuse (objects, templates)	Y	Y
Animation	Y	Y
Real-time viewing	Y	Y
Animation export	N	N

5.6.3 Overall adaptability In a first stage, the intrinsic adaptability of each of the packages was evaluated using the modified adaptability formula described in equation 5.4.4 on page 81. This evaluation was done by implementing the changes listed in section 5.4 on page 79, and applied to the model with and without the additional steps needed for adequate animation purposes (see figures 5.2 and 5.3 on pages 85 and 86 respectively). More specifically, the changes implemented were, in consecutive order:

1 ***add* entity:** new product to basic model
2 ***delete* entity:** Entity 1 from 'change 01' (ie item 1 – add entity)
3 ***delete* entity:** Entity 1 from 'basic model'
4 ***modify* entity:** entity-related delay times
5 ***add* resource:** additional resource (process) to 'change 01'
6 ***add* resource:**
   (a) resource to existing process 'change 05' (ie item 5 – add resource)
   (b) add Work Center to change 05 (Simul8 only)
7 ***delete* resource:** resource from existing process 'change 06'
8 ***delete* resource:** whole process 'change 05'
9 ***modify* resource:** resource in change 01
10 ***add* handling device:** route 1 $\Longrightarrow$ transporter (free) to 'change 01' (Arena only)
11 ***delete* resource & entity:** Prod 2, resource 2, station 2 from basic model
12 ***modify* resource:** transporter $\Longrightarrow$ conveyor in 'change 10' (Arena only)
13 **sequencing:** route $\Longrightarrow$ sequencing in 'basic model'
14 ***delete* hd:** Delete conveyor from 'change 12' (Arena only)

15 ***add* entity:** new product — sequencing (same as 'change 02')

16 ***delete* entity:** Entity 1 from 'change 15'

17 ***add* resource:** as change 05, made to 'change 15'

18 ***delete* resource:** whole process from 'change 17'

19 ***delete* resource & entity:** Prod 3, resource 6, station 6 from 'change 17'

20 ***add* resource:** resource to station 4/Work Center 4 in 'basic model'

21 ***add* schedule:** Schedule for station 4/Work Center 4 in 'change 20'

22 ***delete* schedule:** delete schedule from 'change 21'

23 ***add* breakdown:** add breakdown to resource 6 in 'change 21'

24 ***delete* breakdown:** delete breakdown from resource 6 in 'change 23'

25 ***add* variable:** "Number of entities out" entity 1 in 'change 01'

26 ***add* temporary batch:** process 1 in 'basic model'

27 ***modify* batch:** convert temporary batch to permanent batch in 'change 26'

28 ***delete* batch:** delete permanent batch from 'change 27'

29 ***delete* batch:** delete temporary batch from 'change 26'

30 ***add* permanent batch:** combine 2 products in 'basic model'

31 ***delete* permanent batch:** delete match/combine from 'change 30'

32 ***modify* routing:** reroute 'basic model'

33 ***modify* routing:** rerouting as in 'change 32', implemented in 'change 13'

As figures 5.2 and 5.3 show, both packages are quite adaptable for the same changes. The overall adaptability index with/without animation is 87%/88% for Arena and 82%/86% for Simul8.

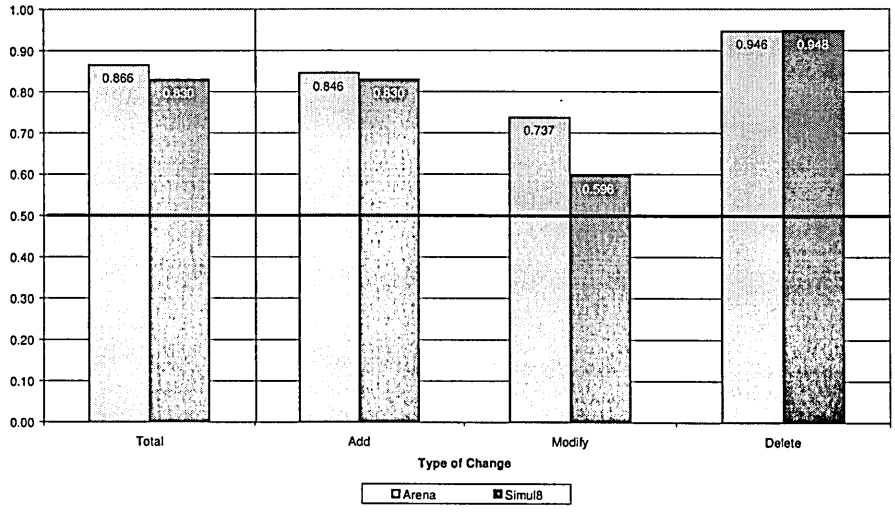

Figure 5.2: Adaptability without animation

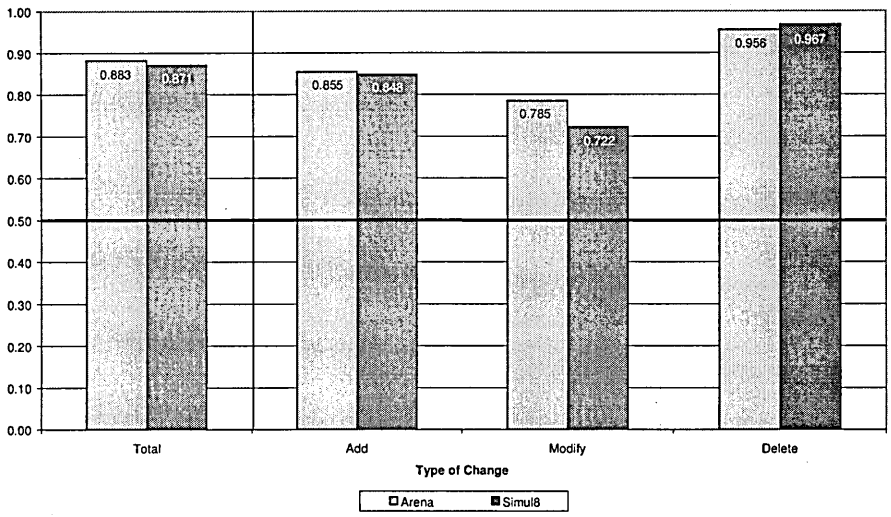

Figure 5.3: Adaptability with animation

However, it can also be gathered from these graphs that Arena™ is overall more adaptable then Simul8™, especially when solely the logic is considered.

This slightly higher adaptability was somewhat surprising to this author, to whom Simul8 felt subjectively easier to handle than Arena, despite better knowledge of the latter product.

When animation is also taken into account, the adaptability of both packages improves, with a much more marked increase in Simul8 than in Arena. This increase in adaptability is due to the fact that creating an animation requires more coding. The additional number of steps needed to create the initial animation is larger than what is needed for subsequent modification, thus resulting in a higher adaptability index for a simulation model with animation than without.

The marked increase in adaptability for Simul8 upon introduction of animation lies in the fact that animation is more closely engrained with this package than it is with Arena. As a result, the effort needed to modify an animation in Simul8 is fairly small compared to the overall effort involved in modifying the whole model. This better integration of animation features in Simul8, also results in the substantial improvement of the adaptability, compared to Arena.

Also, in both cases the adaptability coincides quite well with the results obtained from the questionnaire survey (see § 4.2.4 and figure 5.4 on the following page). The benchmarking suggests that deleting elements (i.e. entities, resources/processes and handling

devices) is the easiest, and modifying them the hardest. From the user survey it emanates that, when asked as to what is the easiest change to implement, *deleting* was ranked top and *modifying* as the most difficult among the three.

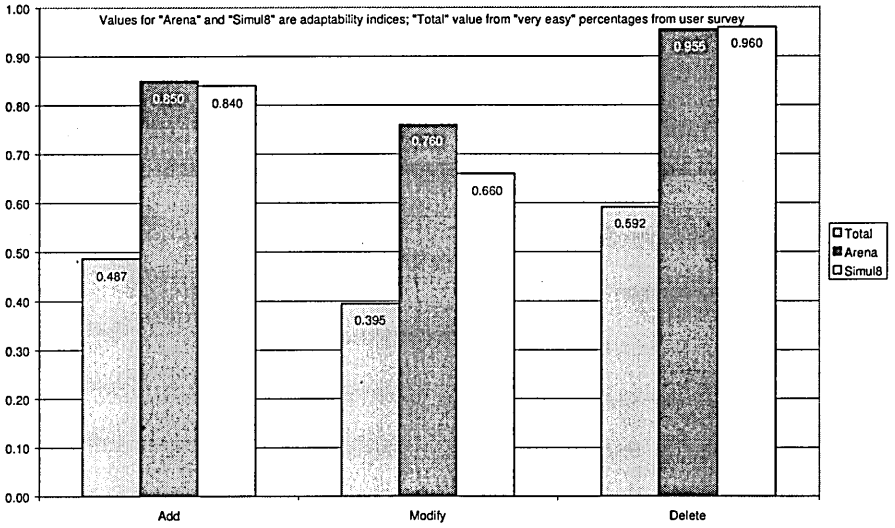

Figure 5.4: Adaptability (with & without animation averaged)

5.6.4 Sequencing Figure 5.5 shows the effect of sequencing on the overall adaptability of simulation models. The basis for this graph was the addition of a work centre (process) to an existing simulation model using hard coding (left) and sequencing (right).

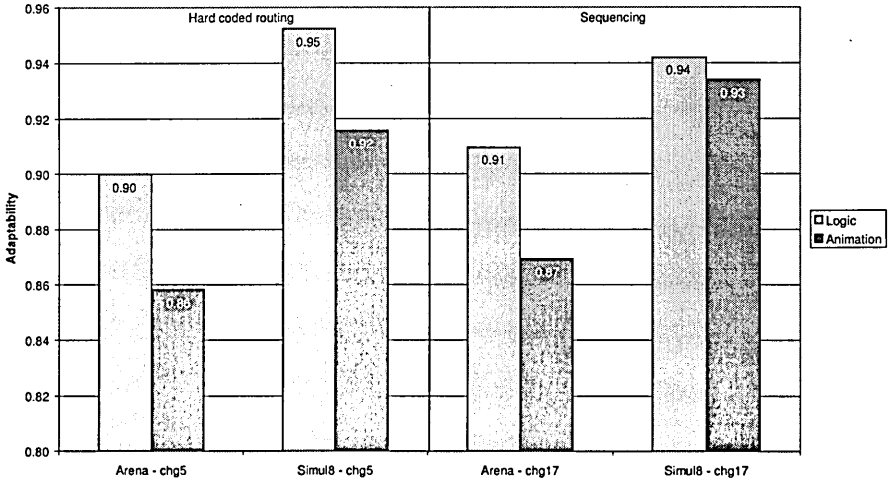

Figure 5.5: Influence of sequencing on adaptability

On its own, this graph is very inconclusive, suggesting an increase in adaptability for Arena of only 0.01 and no change for Simul8 when the adaptability for the model with

and without the animation is averaged (see table 5.2). The graph does, however, show that Simul8 seems to be overall more adaptable than Arena.

Table 5.2: Adaptability values for Arena and Simul8

Software	Without animation	With animation	Average
Arena – hard coding	0.900	0.858	0.879
Simul8 – hard coding	0.952	0.916	0.934
Arena – sequencing	0.910	0.869	0.889
Simul8 – sequencing	0.942	0.934	0.938

However, when looked at by category (i.e. addition, modification and deletion of elements such as entities and resources), a different picture emerges. Figure 5.6 gives the adaptability for Arena and Simul8, averaged with and without animation. This graph indicates that, by using sequencing, the overall adaptability improves for both packages when adding or modifying of elements is concerned, while there is little change in adaptability for the deletion of elements. More specifically, there is a marked increase in adaptability for Simul8 when sequencing is used for modifying elements, while the increase is less marked for Arena. Both packages increase somewhat in adaptability for the addition of elements when sequencing is used, compared to hard coding. For the deletion, however, there is no change for Arena while the adaptability, when using sequencing, even decreases slightly for Simul8.

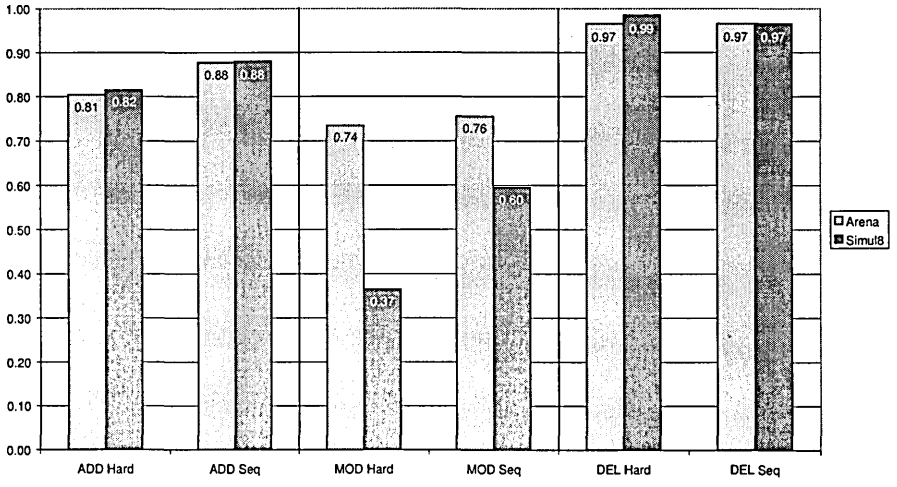

Figure 5.6: Adaptability of hard coding and sequencing by category

By calculating the adaptability coefficient $C_{(A)}$ of hard coding versus sequencing (see equation 5.6.1 on the following page), the increase or decrease of the overall adaptability

by using sequencing instead of hard coding becomes more obvious, as can be gathered from figure 5.7.

$$C_{(A)} = \frac{A_{(\text{sequencing})}}{A_{(\text{hard coding})}} \tag{5.6.1}$$

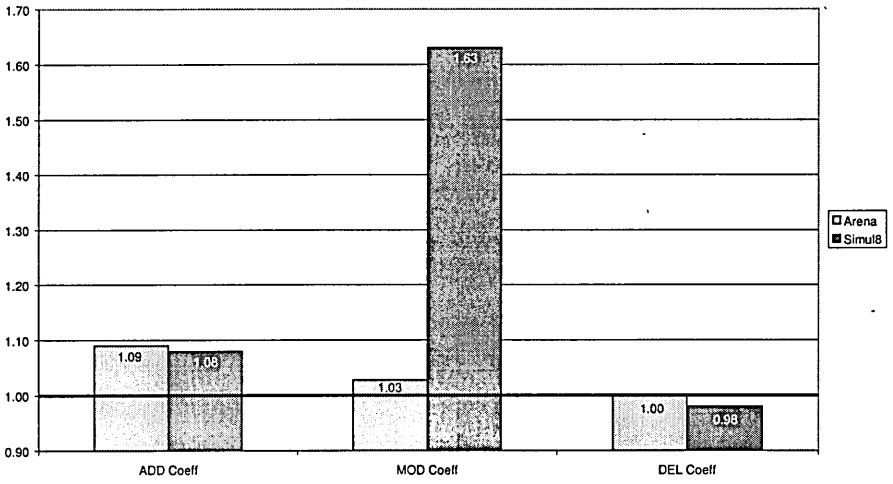

Figure 5.7: Adaptability coefficients

As figure 5.7 indicates, there is a clear net benefit to be gained from using sequencing, rather than hard coding, in each COTS package. Simul8 gains more from sequencing than Arena, with an averaged difference to $C_{(A)}$ of $\frac{1.08+1.63+0.98}{3} = 1.23$ for Simul8 and $\frac{1.09+1.03+1.00}{3} = 1.04$ for Arena.

5.6.5 Organizational layout A total of 19 people participated in this experiment. The participants were mainly postgraduate, but also some undergraduate, students from the TU Delft (Netherlands) and Sheffield Hallam University. Figures 5.8 and 5.9 show the animation and different layouts for the initial model, which was then modified according to the specification outlined below.

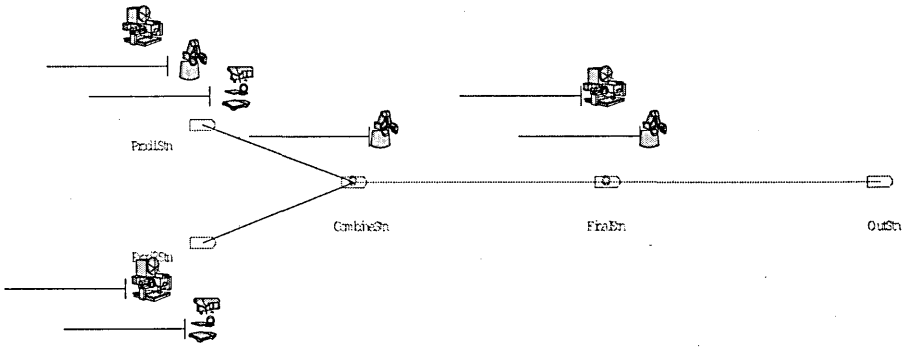

Figure 5.8: Model animation

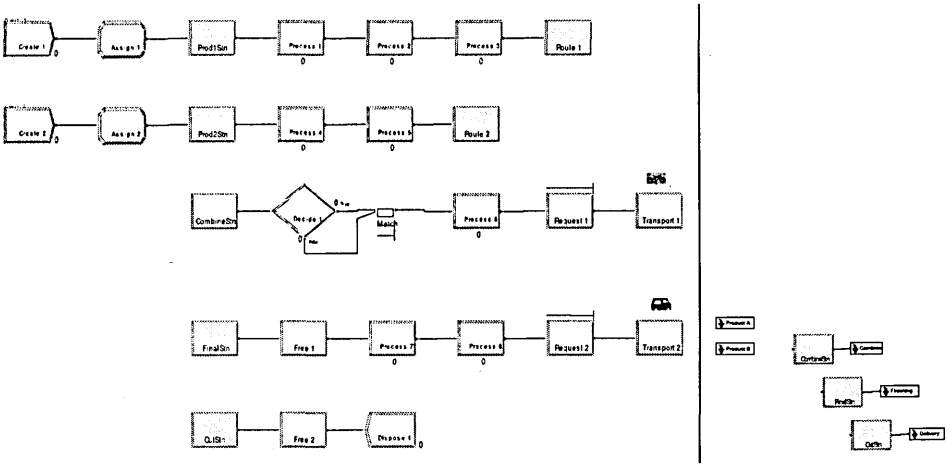

Figure 5.9: Model logic without (L) and with sub-models (R)

The modifications to be implemented were:

1 *Delete the third process from Product A (in Station Prod1Stn)*
2 *Create a third product, Product C, with interarrival times of EXPO(4) and Process Times of*
   (a) *Process 6: NORM(2,0.06)*
   (b) *Process 10: EXPO(3.7)*
3 *Create an additional station for product C, FinalCStn with process 10 (seize-delay-release create a new resource Res10machine) and routing (via conveyor) to Delivery (i.e. OutStn) see task 4 for conveyor*
4 *Include an additional transporter to transport elements from CombineStn to FinalCStn (distance: 20; load/unload times insignificant)*

5 *Include an additional conveyor to transport elements from FinalCStn to OutStn (distance: 20; load/unload times insignificant)*

6 *Increase the capacity of Resource 6 (Res6Machine) from 1 to 2*

7 *Exchange transporter 2 for a conveyor (distance remains the same, load and unload times are insignificant, i.e. do not need to be modelled)*

After making those modifications, the candidates were asked to fill in a short questionnaire, rating the difficulty of the tasks (i.e. adding, modifying or deleting entities, resources or handling devices) from *[very easy]* to *[very difficult]*. They also had to rate the subjective helpfulness of the structure from *[very helpful]* to *[not at all]* helpful.

In order to avoid experience-based bias in the results, the candidates were also asked to qualify their abilities and experience with Arena from *[poor]* to *[fair]*, *[good]* and *[very good]*. — The complete task description and questionnaire form can be seen in *Appendices C* and *D*.

The experience-weighted results can be seen in figure 5.10.

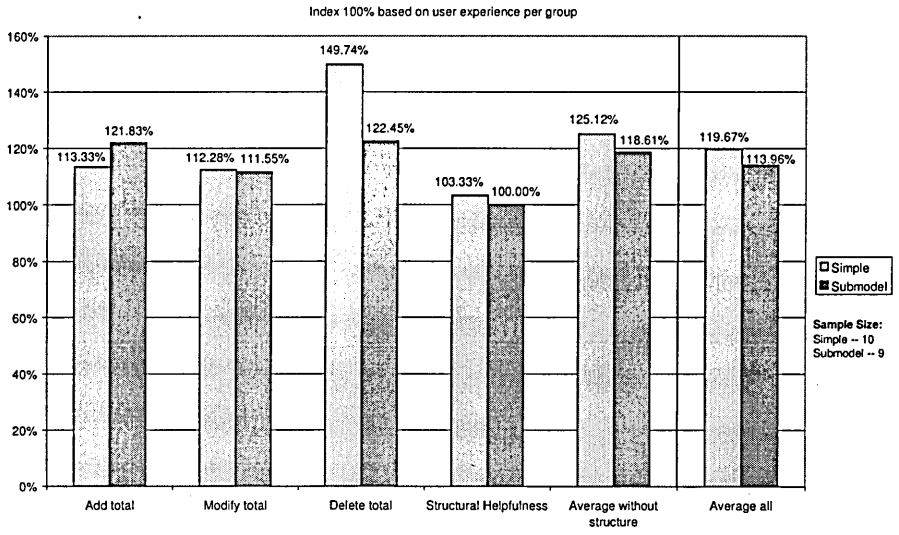

Figure 5.10: Influence of use of sub-models on overall ease of use

As these data show, the level of difficulty to change various elements, as obtained from the original questionnaire survey and the benchmarking of the overall adaptability, has been reconfirmed. Adding elements has yet again resulted as being the easiest task, while modifying them remains the most difficult.

However, the results also show that performing a deletion or modification was easier in the simple model than in the one with sub-models, and only adding elements was

easier when sub-models were used. Overall, modifications in the simple model were rated substantially easier.

This suggests that, while the use of sub-models simplifies the primary level, thereby hiding the elements that actually need changing in lower levels, this makes it more difficult for a simulationist other than the creator of the original model, to implement any type of change.

As the simple model was laid out in a very similar fashion to the one with sub-models, i.e. geographically grouping elements belong together, it can be suggested that, for the ease of future modifications, no elements such as sub-models ought to be used. The same also – and especially – applies to customized templates, when changes would have to be made to the internal parts of those templates.

This suggestion is further substantiated by comments from the participants in this benchmarking exercise:

> *"It takes me more time to understand a model completely when it's built up with sub-models. I easily lost my overview with this model."* — [data05]
>
> *"Deleting or modifying is not the hard part. Repairing the damage is where it gets tricky. For a relatively straight-forward model like this it's ok, for a complex one it could get very nasty, and would require a much deeper insight to the model."* — [data08]
>
> *"The submodel interface is not very helpful. If there was a function to split the screen into two parts, one for the submodel and the main window it would be better."* — [data10]
>
> *"The sub-models are helpful for understanding the structure but for working with the model disturbing."* — [data11]

As these quotes show, the use of sub-models – even in a small and rather simple model as the one used for this study – complicates the implementation of modifications, makes tracking of bugs harder, and should therefore be avoided when future modification of the model by someone else is likely.

5.7 Key findings

The benchmarking exercise first of all confirmed the findings from the questionnaire survey, showing once again that deleting elements (entities, resources, handling devices) is the easiest, and modifying elements the most difficult task, and that any type of change to entities is the easiest, and changes affecting handling devices the most difficult.

It also revealed that, when incorporating animation into the model, the overall adaptability slightly increased, i.e. the additional effort to modify the animation is less than the overall effort needed to modify the model logic.

Overall, Arena™ has a slightly higher adaptability rating then Simul8™, even though Simul8 *felt* easier to handle. The advantage of Arena over Simul8 was most marked for adding and modifying elements, both with and without animation, while Simul8 has a slight advantage over Arena when it comes to deleting elements, and that only if the animation is included.

When it comes to sequencing, which is generally considered helpful when building large(r) models, Arena is somewhat better for adding or deleting elements, but Simul8 has a substantial advantage over Arena for modifying elements that are connected to sequences.

Most COTS packages offer some form of grouping elements together, such as the creation of templates or submodels. While this is considered helpful in building and organizing a simulation model, the controlled experiment showed that the use of submodels, which was used in this case, complicates understanding of a model when it is passed on to someone else for modification. This means that, when a model is built with the intention of later passing it on to someone else, layered grouping should be avoided, and graphical grouping used instead.

Chapter 6

Modified Model Building Guidelines

This new methodology is closely based on the *"Steps in a simulation study"* by Law and Kelton (2000); see also § 2.3.1.

While the methodology identified in the stage 1 interviews (see section 4.1), as being closest to what the practitioners used, was the one developed by Banks and Carson (1984), it seemed less adequate for the purpose envisaged here. The purpose for which the guidelines shown below were created, was to enable infrequent/irregular users of COTS-packages to implement changes to an existing, once shelved, simulation model more easily, so as to prevent them from having to create a new model from scratch. As previous chapters indicate, there is substantial reluctance to modify existing models, as they tend to be seen as too complex to understand by someone other than the original model builder. Those guidelines are mainly aimed at infrequent/irregular users of COTS-packages as these are most likely to use some modelling guidelines, some form of help in written form, to build a model. The use of the guidelines described below should further enable them reuse such models at a later stage, thereby increasing the use of simulation in areas where there is, so far, a certain reluctance to do so.

6.1 An overview

The methodology has been modified so as to include improved adaptability of simulation models when they are built from scratch. All modifications to the actual steps listed below are clearly marked in ***bold italics***. See figure 6.1 on page 98 for a graphical representation.

1 Formulate the problem and plan the study
   (a) Problem of interest is stated by manager

(b) One or more kickoff meetings for the study are conducted, with the project manager, the simulation analysts, and subject-matter experts (SMEs) in attendance. The following issues are discussed:

- Overall objectives of the study
- Specific questions to be answered by the study
- Performance measures that will be used to evaluate the efficacy of different system configurations
- Scope of the model
- System configurations to be modeled
- Software to be used
- ***Anticipate and plan for future modifications of model – through inclusion of higher management*** **and** ***SMEs***
- Time frame for the study and the required resources

2 Collect data and define the model

(a) Collect information on the system layout and operating procedures

- No single person or document is sufficient
- some people may have inaccurate information – make sure that true SMEs are identified
- Operating procedures may not be formalized

(b) Collect data (if possible) to specify model parameters and input probability distributions

(c) Delineate the above information and data in an "assumptions document," which is the *conceptual model.*

(d) Collect data (if possible) on the performance of the existing system (for validation purposes in Step 6)

(e) The level of model detail should depend on the following:

- Project objectives
- Performance measures
- Data availability
- Credibility concerns
- Computer constraints
- Opinions of SMEs
- Time and money constraints

(f) There need not be a one-to-one correspondence between each element of the model and the corresponding element of the system

(g) Interact with the manager (and other key project personnel on a regular basis)

(h) ***Identify possibilities for future modification/reuse of the simulation model***

3 Is the conceptual model valid?

(a) Perform a structured walk-through of the conceptual model using the assumptions document before an audience of managers, analysts, and SMEs

- Helps ensure that the model's assumptions are correct and complete
- Promotes ownership of the model
- Takes place *before* programming begins to avoid significant reprogramming later

4 Construct a computer program and verify

(a) Program the model in a programming language (e.g., C or FORTRAN) or in simulation software (e.g., Arena™, AutoMod™, Extend™, ProModel™, WITNESS).

Benefits of using a programming language are that one is often known, they have a low *purchase* cost, and they may result in a smaller model execution time. The use of simulation software, on the other hand, reduces programming time and results in a lower *project* cost.

(b) ***Follow adaptability guidelines to provide for future reuse and modification of model*** (See section 6.3 on page 102 for details).

(c) Verify (debug) the simulation computer program.

5 ***Does the programmed model provide for potential future modifications?***

(a) ***Do the adaptability options provide sufficient scope for later modification/-reuse of the model?***

(b) ***Does the programmed model, in terms of potential adaptation, tie in to the company's – or department's – mid- to long-term strategy/plans?***

6 Make pilot runs

(a) Make pilot runs for validation purposes in Step 7

7 Is the programmed model valid?

(a) If there is an existing system, then compare model and system (from Step 2) performance measures for the existing system

(b) Regardless of whether there is an existing system, the simulation analysts and SMEs should review the model results for correctness

(c) Use sensitivity analyses to determine what model factors have a significant impact on performance measures and, thus, have to be modeled carefully

8 Design experiments

(a) Specify the following for each system configuration of interest:

- Length of each run
- Length of the warm up period, if one is appropriate
- Number of independent simulation runs using different random numbers – facilitates construction of confidence intervals

9 Make production runs

(a) Production runs are made for use in Step 9

10 Analyze output data

(a) Two major objectives in analyzing output data are:

- Determining the absolute performance of certain system configurations
- Comparing alternative system configurations in a relative sense

11 Document, present and use results

(a) Document assumptions (see Step 2), computer program, and study's results for use in the current and future projects.

- ***Document all assumptions relating to future modification/reuse of the model.*** – The assumptions made for the initial simulation might not hold up for the modified one, and should therefore be clearly stated in order to prevent the modified model from being inaccurate.
- ***Create detailed documentation of computer program*** – e.g. of individual submodels or individual modules – to ease potential problems due to a change in modeler. If otherwise structures are unclear, a printout of the modules showing the logic connectors, or a flowchart, should be included.

(b) Present study's results

- Use animation to communicate model to managers and other people who are not familiar with all of the model details.
- Discuss model building and validation process to promote credibility

(c) Results are used in decision making process if they are *both* valid and credible

(d) ***Document current simulation modeller/users and establish responsibilities for future projects***

- ***Which department holds documents and model***
- ***Prepare for current knowledge owner's absence, i.e. establish order of command/order of reference***

(e) ***Specifically document anticipated modifications and their scope to avoid unnecessary confusion when changes are to be implemented in the model***

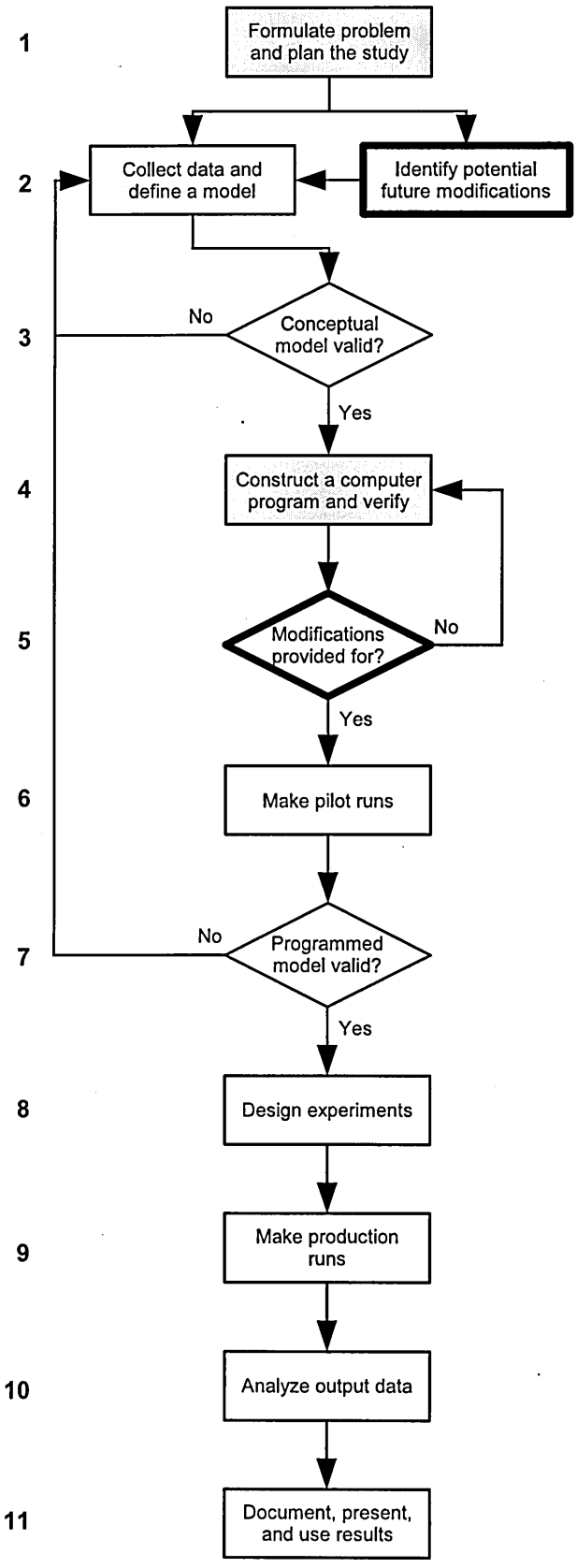

Figure 6.1: Modified model building guidelines, based on Law and Kelton (2000) — modified elements underlayed in grey, new elements with thick border

6.2 The guidelines in detail

In this section, the elements added to the guidelines are looked at in detail.

6.2.1 Formulate problem and plan the study *Step 1* in figures 2.1 on page 9 and 6.1 on the preceding page [in *item 1.(b)* in the list on page 94] has been modified to include planning for future modifications by including higher management *and* subject-matter experts.

Traditionally, only the management directly affected by the simulation is involved in the whole process. Law and Kelton (2000), Balci (1990), Banks and Carson (1984) and Kelton et al. (2002) all point out the importance of involving the project manager and the management of the area affected by the simulation throughout the course of the study. However, as medium- to long-term strategy has a substantial influence on the whole company, and mainly senior management being involved in such medium- to long-term decisions, the author believes it is important for such higher management to be directly involved in, at least the initial stages of, the simulation study.

Another potential benefit of including higher management is that thereby they, too, assume some ownership of the simulation study, thus becoming more interested. I.e., this is a potential for improving the spread of simulation.

6.2.2 Identify potential future modifications *Step 2* (box on the right) in figure 6.1 on the previous page [*item 2.(h)* in the list on page 94] has been added, as the original methodology proposed by Law and Kelton (2000) does not look at the potential of future modifications.

As pointed out by Yilmaz and Ören (2004), Spiegel et al. (2005) and Garlan et al. (1995), one of the main issues affecting the potential for software and -component as well as model reuse is the lack of explicitness of assumptions and constraints. In the case of potential future model modification, not only general assumptions should be made, but also assumptions regarding the type of changes to be anticipated in the future. If an organization's long-term strategy points to, e.g., diversification, then the potential influence of that strategy on the area under current study should be looked at and relating modifications planned for. Such assumptions that ought to be taken into account can be, eg the addition of further products, or a change of the production from a push to a pull system.

By concurrently defining the model *and* identifying potentials for modifications of the model that is being built, the requirement to make assumptions and constraints explicit is brought closer to realisation. As Spiegel et al. (2005) pointed out, capturing *all* assumptions and constraints is very unlikely to happen.

Also, as Kulick and Sawyer (1999) stated: *"The primary objectives of any simulation effort often determine the complexity of the tools needed"*. It is therefore of utmost importance to be aware, and plan for, future modifications, as this can – and will – impact on the complexity of the model.

6.2.3 Construct a computer program and verify In order to incorporate the software-based provisions for adaptability, it is important to establish and follow specific adaptability guidelines. The modifications to *step 4* in figure 6.1 on page 98 [*item 4.(b)* in the list on page 94] takes this into account.

Section 6.3 treats those necessary elements to provide for improved adaptability in detail.

6.2.4 Provisions for modifications *Step 5* in figure 6.1 on page 98 [*item 5* in the list on page 94] has been added as a separate step, as verifying whether the coded model provides for future modifications beyond the scope of the planned simulation study is an issue independent from creating a working and accurate computer model. That is, making sure the programmed model allows for future modification is a question of validity, rather than of verifying/debugging the code.

The verification of the provision for adaptability consists of two steps, which should – again – be done concurrently:

1 Comparison of the programmed model against general adaptability guidelines
2 Comparison of the adaptability coded into the model against the medium- to long-term strategies of the organization under study. — This differs from *item 1.(a)* in so far, as *1.(a)* is only concerned with the short-term strategy, i.e. the immediately to be achieved goal of conducting a simulation experiment.

Although listed as a separate logical step, this verification would be done concurrently with the coding of the model.

6.2.5 Documentation, presentation and use of results *Step 11* in figure 6.1 on page 98 (*step 10* in figure 2.1 on page 9 and *item 11* in the list on page 94) had to be greatly modified and extended to accommodate potential future adaptations.

Document all assumptions relating to future modification/reuse of the model: The documentation as suggested by Law and Kelton (2000) only contains assumptions relating to the current study. However, in order to facilitate reuse of the model at a later stage, it is important that all further assumptions, which were made to incorporate adaptability (if there was a need for any such additional assumptions) are also documented. This relates again to the problem pointed out by Yilmaz and Ören (2004), Spiegel et al. (2005) and Garlan et al. (1995).

Create detailed documentation of computer program – e.g. of individual submodels or individual modules – to ease potential problems due to a change in modeler: Although documentation of the computer program forms an integral part of the Law and Kelton (2000) approach, the adaptability-specific parts of the model should be additionally indexed, or included in a separate documentation, so that, if a new study based on a modification of an existing model is to be conducted, the modeller(s) can quickly evaluate the fit of the model to the new circumstances. For this reason, the documentation should include a flowchart-like representation, if a printout of the model or submodels does not offer sufficient information in terms of structuring and the flow of elements or sequence of processes.

Although Law and Kelton (2000), Balci (1990), Banks and Carson (1984), Kelton et al. (2002), Benjamin et al. (2000), Robinson (1994), Tye (1999) and de Vreede et al. (2003) pointed out the importance of documenting the simulation model, none of these authors go into detail about how to document the computer program.

Gass (1978), among others, stated that "*documentation should commence at the very beginning of a project*", and that "*the 'lack' of documentation is one of the main reasons cited for model failures*".

Gass also suggests a complete documentation methodology, but is more specific about the documentation of the program itself. He suggests to document, among others:

- input and output data, and test case that have been run
- flow charts of the program
- operating instructions for the computer operator
- explanation of options available in the model

- a printout of the actual program code

In addition, to these documentation principles put forth by Gass, documentational aids provided by the program itself should be used. Simul8™ for example allows the addition of comments to most, if not all, elements, and Arena™ offers this option on, e.g. submodels.

Document current simulation modellers/users and establish responsibilities for future projects: While simulation projects are generally conducted within a team, all the team members will have access to the model and documentation. However, this step in the documentation is relevant when the team is very small, and/or to provide for the possibility that another, unrelated team, may have a use for the simulation project.

Specifically document anticipated modifications and their scope to avoid unnecessary confusion when changes are to be implemented in the model: This differs from *item 11.(a)* in so far as this is not concerned with assumptions, such as simplifications of certain input data. The anticipated modifications, in this case, refer to the changes or types of changes identified as the most likely ones in step *1.(b) – Anticipate and plan for future modifications of model, through inclusion of higher management and SMEs.*

6.3 Adaptability guidelines

The adaptability guidelines explained in this section are based on the results from the questionnaire survey (section 4.2 on page 64) and the benchmarking (section 5 on page 74) and influenced by the interviews conducted at an earlier stage (section 4.1 on page 61).

Adaptability of simulation models for implementation of more substantial changes, several months or years after its last use, can be greatly aided by using the following upon building the initial model:

1 Sequencing
2 Model layout
3 In-built annotation possibilities
4 Type of data (internal vs external)

6.3.1 Sequencing The use of sequencing may, at first, increase the complexity of the model somewhat (i.e. increase the number of steps needed to build the model), but has shown to simplify the adaptability of the model later on. This is because, contrary to

hard-coding of all information, sequencing allows the user to specify the routes entities take in a more centralized way, e.g. by means of an internal database.

When hard-coding is used, for example after each resource where there are different directions an entity can take, to determine where that entity should go, many decision blocks have to be added. Those decision blocks would state that *"if the entity is of type* A, *then it should go to the left, otherwise, to the right"*, and one such block would be needed at every decision point.
When sequencing is used, however, these blocks need not be used, as per item there will be a list of resources, or work stationgs, that item has to go through. This means that some form of database, within the simulation package itself, is populated with data stating, for example, that *"entity of type* A *has to go through work stations* 1, 2, 3, 5 *and* 7, *while entity of type* B *has to go through work stations* 1, 3, 4, 6 *and* 7*"*. As a result, the person in charge of modifying the model only needs to look in one place to find out where elements are going, rather than having to look through the whole code.

However, as sequencing can cloud the flow of items or processes, a flowchart should be included as part of the documentation.

6.3.2 Model Layout Contrary to sequencing, which does decrease the number of programming steps at the time of implementing a change, the use the model layout has no such direct benefits. However, along with annotation, it helps the user understand and navigate the existing model, thereby reducing the time needed to understand and consequently modify the model. This, in turn, cuts down on the overall model-building and -modifying time and reduces the perceived need to build a new model from scratch.

It is best to avoid complex, tiered structures, such as submodels or templates. While they are helpful in reducing the number of elements visible at any one time, and therefore aid the basic, high-level understanding, they make it more difficult for someone other than the original simulation builder to modify the model. Graphical aids, e.g. grouping objects and using graphical tools, such as drawing boundaries around groups, also help in gaining a primary, high-level understanding, while at the same time granting uncomplicated low-level access, and should therefore be given priority over submodels (and/or custom-built templates in packages such as Arena).

Tiered structures result in the model consisting of different layers of varying detail. The top-level would only show the connection from submodel to submodel, and the user

has to open the submodel to see how elements are routed within it, or what resources and handling devices there are. By only graphically grouping everything, for example by drawing a square around the modules that make up one work center, this helps the user to still differentiate between the different logic groupings, while still enabling that user to see all the connections which would otherwise be hidden.

Where available, *named views* (e.g. in Arena) should be used, as these also help in navigating the model. With named views, one specific area of the modelling surface, e.g. what would amount to one subgroup, can be assigned a view, to which the user can switch from, e.g. the overall model view.

6.3.3 Annotation While submodels help organize the model in a logical way, thereby making it easier for the user/model builder to navigate through the simulation model, the annotation facilities built-in to most modern COTS simulation packages can be of great help in clarifying ambiguous terms (e.g. the use of specific variables can be explained in more detail or confusion prevented).

Where there is an option to output actual code (such as in Arena, with the *"write SIMAN"* command), this should also be added to complete the documentation/annotation.

Another highly important element to help with annotation – and understanding of the model itself – is the choice of names. All resources, attributes and variables should be given logic names that are easy to understand; e.g. attribute names such as *att1*, *att2* etc. should be avoided in favour of names such as *itmlgth* for the attribute *item length*. Also, attributes, variables and resources, as well as other elements of significance, should be stated as part of the documentation, with a short description, e.g. the name and what the element represents.

As mentioned in section 6.3.1 on page 102, a flowchart of the processes should be included, especially if sequencing is used, as this can make understanding the structure more difficult.

6.3.4 Data type Depending on the (estimated) amount of input data, the simulationist has to decide whether to include that data in the simulation (e.g. incorporated into the various elements), or put it into an external database (e.g. a database such as Microsoft's *Access* or a spreadsheet, such as Microsoft's *Excel*), as a later change from internal to external tends to be quite cumbersome. This approach works well for simulation models

with a large amount of input (or output) data, especially if that data already happens to be present in the form of a database or spreadsheet.

Kulick and Sawyer (1999), talking about their own experience, suggest that *"for ease of maintenance and flexibility, the non-simulation model components are all implemented using the common programming framework of Microsoft Visual Basic for Applications (VBA)."*

However, the use of externally stored data, e.g. in the form of a database, requires some VBA (or other programming language) knowledge and a good understanding of the simulation package.

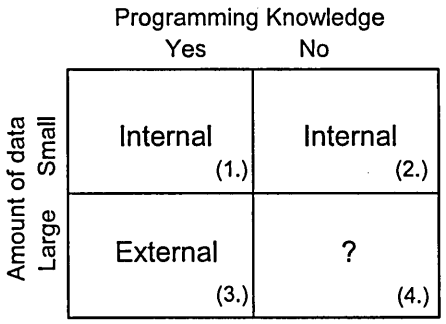

Figure 6.2: Data placement matrix

The matrix in figure 6.2 is intended to aid in deciding whether or not to use the data – including estimates for future data requirements – internally or externally. In this case, *"small amount of data"* refers to the amount of work needed to include the data internally being substantially smaller than that needed to store the data externally and programming the interface. This *'amount of work'* could for example be measured in man-hours. As what constitutes 'large' or 'small' amounts of data is very subjective, no clear cut-off point for its placement within or outside the simulation model can be given at this point.

The matrix (fig. 6.2) in detail:

1 Small amount of data and programming knowledge: No need for storing data externally
2 Small amount of data, but no programming knowledge: No need for storing data externally
3 Large amount of data and programming knowledge: Use of external data storage strongly suggested, even though this means that any future model handler, who has to modify it, will need to have those specific programming skills

4 Large amount of data, but no programming knowledge: Use of external data storage still suggested, but only if model handler is willing and able (abilities as well as time allowance) to learn the necessary skills

However, it also has to be noted that, in the case of externally stored data, the requirements for documentation also increase; specifically, the documentation will have to include:

- Details about the external storage method (name and description of database/spreadsheet, caveats)
- Details about the interface (to enable modeler to understand and retrace the steps made during the first build phase and modify the code if and where necessary)

Chapter 7

Validation of the Guidelines

This chapter explains how the modified model building guidelines, introduced in chapter 6, are tested and validated.

7.1 Validation interviews

The aim of the validation interviews was to get feedback on the provisional *model building guidelines* before completing and testing them.

While at first it was planned to conduct such validation interviews with a few people, only one such interview was held, due to time constraints and general unavailability of interviewees with sufficient experience within reach.

However, the interviewee chosen has many years' worth of experience as simulation consultant, whose views and opinions are of immense value.

The feedback thus obtained therefore provided excellent, highly valuable and helpful results.

7.1.1 Interview questions The validation interview questions were grouped into 3 sections; the first one on the interviewee's simulation experience, as a user's experience was considered of high value in giving feedback on the guidelines.

The second section concerned the overall *model building guidelines*, which are based on the *steps in a simulation study* by Law and Kelton (2000), see section 2.3.1 and chapter 6 for details, while the third section only looked at the *adaptability guidelines*, i.e. a subset of the overall *model building guidelines*. (See also Appendix E).

Interviewee background

1 How many years experience do you have in simulation?
2 Which software package(s) do you mainly use?
3 How many projects have you approximately completed? (per software)
4 What is, approximately, the average duration of such projects?

The interviewee's experience, as indicated above, has great influence on the amount and type of feedback they can give, hence the first and third questions of this section.

The second question aims at identifying the type of software package, as the use of, e.g. a pure programming language, would be of limited value, as the *model building guidelines* are mainly meant for people who use COTS packages.

The average duration of a simulation project is important, as, if the user only conducts very small simulation projects, they might not be aware of some issues affecting larger models, such as the need for grouping data in different ways, or the tendency to use tiered structures.

Questions on model building guidelines

1 What is your general opinion about those guidelines?
2 Is the graph *Data placement matrix* easy to understand?
3 Is the anticipation of future potential modifications a good idea?
4 What do you think about the inclusion of higher management and/or the long term strategy?
5 Do you believe your COTS has sufficient internal annotation options?
6 Is the preparation for future reuse by someone else helpful?
7 Is the documentation of owners and keepers of further data and responsibility thereof good idea?
8 Is the documentation of anticipated modifications (i.e. company strategy) good idea?
9 Is there anything missing, incomplete or superfluous?

All the above questions are meant as introductory, opening questions, from which further questions could then ensue.

Questions on adaptability guidelines

1 What is your general opinion about the adaptability guidelines?
2 What is your general opinion on Sequencing, submodels, annotation and data types, as outlined in the guidelines?
3 Is there anything missing, incomplete or superfluous?

As mentioned for the previous paragraph, the questions in this group were also just opening questions, which could then lead on to further, probing questions.

7.1.2 Interview feedback The interview itself was conducted in two stages; a preparatory stage and the actual interview. The preparatory phase consisted of explaining the background of the *model building guidelines* to the interviewee, and handing him a copy of those.

During the second stage, conducted a few days later, the questions stated in section 7.1.1 were answered and those answers then explored further.

Interviewee background – answers

1 Approximately 9 years experience in simulation
2 Mainly use Arena, but also sufficient knowledge of Automod, Promodel, eM-Plant and EnterpriseDynamics
3 Have completed more than 40 projects.
4 Short-term projects take roughly 1 month, medium-term projects approximately 2 – 3 months and long-term projects typically take between 6 and 12 months.

Thus, the interviewee in this case has extensive experience in simulation, mainly with COTS packages, out of which predominantly with Arena.

Questions on model building guidelines – answers

1 Open door; everything changes in a simulation project anyway, and these guidelines make everything seem more complex than it is. Item *5.(b)* in the guidelines seems identical to item *1.(a)*, and item *11.(c)* seems identical to *11.(a)* – documentation of assumptions.
2 yes

3 yes

4 yes

5 yes

6 yes

7 Yes; however, as simulation projects are mostly done in small groups, within a department, that team will always share that project anyway. Also, preparing for handing a project over, or keeping track of owners/responsibilities never happens.

8 yes

9 yes

Overall, the interviewee felt that any additional help from the model building guidelines presented to him would be doubtful in a real simulation project, and that a number of items in the guidelines were superfluous. He expressed his opinion that some elements, eg regarding the documentation, were already implicit in the *steps in a simulation study* by Law and Kelton (2000) .

Questions on adaptability guidelines – answers

1 vague and unclear

2 Sequencing is best used in random systems, not in semi-random systems; It is good for within subsystems, but can make it more difficult to understand the model if sequencing is also used *between* subsystem, rather than solely *within* them.

3 yes – data structures; one needs to distinguish between packages where processes are defined (e.g. Arena) vs packages, where resources, rather than processes, are the focal point (e.g. Simul8).

Overall, the interviewee felt that the specific adaptability guidelines would not enable, or improve, further reuse.

7.2 Experiment

As stated in sections 3.4 and 3.5.3, on pages 57 and 60 respectively, a *framed field experiment* was conducted with a test- and a control group, in order to identify the helpfulness of the adaptability guidelines.

7.2.1 Test groups Both groups were given data and requirements to build a simulation model, which they then had to pass on to a third group. This third group then had to implement identical changes in both sets of models, and rank the difficulty of performing those changes.

The test group was also handed the full set of adaptability guidelines, while the control group had to rely on their knowledge, experience, and generally available literature on the subject.

As the model had to be representative of the type and size of simulation models normally encountered in the real world, the requirements were for rather large models, compared to normal university/student models.

This resulted in substantial difficulty in finding a larger sample size, and thus this experiment was conducted with only three people with extensive modelling experience; one test person, one control person and one person to make the modifications and rank the difficulty of implementing them.

Due to the requirement to test the guidelines in a natural setting, the experiment could not have been extended to include students.

7.2.2 The models As mentioned above, the experiment consisted of handing initial data and model requirements to practitioners to build the model.

In order to keep the model within realistic limits, those requirements and data were based on a real-life simulation model in the area of manufacturing.

Also, in order to test the guidelines and their usability against a widest possible range of modifications, data and requirements were chosen that would result in the implementation of a substantial part of the changes listed in section 5.4.

Initial data & requirements

The initial data for the simulations can be seen in Appendix F (*Validation Simulation – Part A* and *Validation Simulation – Part B*)

Part *B* also included the complete set of model building guidelines, as shown in Appendix G

Modifications

Once the initial simulation models were completed, they were passed on to the third group, which then implemented certain changes affecting:

- work schedules
- entities (inter-arrival times)
- attributes
- resources (modifications and additions)
- various time-related settings (delay times, break-down times etc.)
- handling devices
- variables

The completed models from the initial modellers (parts *A* and *B*) were passed on in their electronic form (i.e. as **.doe*-files) to the third modeller to implement those modifications.

In addition to the electronic files and data/requirements for the modifications, the following was also submitted to the modifier:

- data and requirements for the initial models
- comments from the initial model builders (as shown in Appendix F, on pages *F-i* and*F-iii*.)
- printouts of the model code documents (**.opw*-file defining resources and variables, **.mod*-file, containing the actual logic and **.exp*-file containing the model data, e.g. attributes, variables...). See Appendix F, pages *F-iii* and *F-v*, for details.
- copy of *model building guidelines*

The full set of requirements and data for these modifications can be seen in Appendix F (*Validation Simulation – Part C*)

The modifier implemented the required changes to both simulation models (i.e. parts *A* and *B* simultaneously), and ranked them as *very easy*, *somewhat easy*, *somewhat difficult* or *very difficult*.

Those rankings were then assigned the values 1 *(very difficult)* – 4 *(very easy)*. From these numerical rankings, an overall adaptability factor was then calculated for the ease of modification for both models.

The resulting adaptability factors for adding, modifying and deleting elements, as well as the overall adaptability factor, are shown in table 7.1 on page 113 and figure 7.1.

Table 7.1: Adaptability factors

	Part A	Part B
Total	1.68	3.43
Add	2.00	3.14
Modify	1.67	3.57
Delete	1.33	3.57

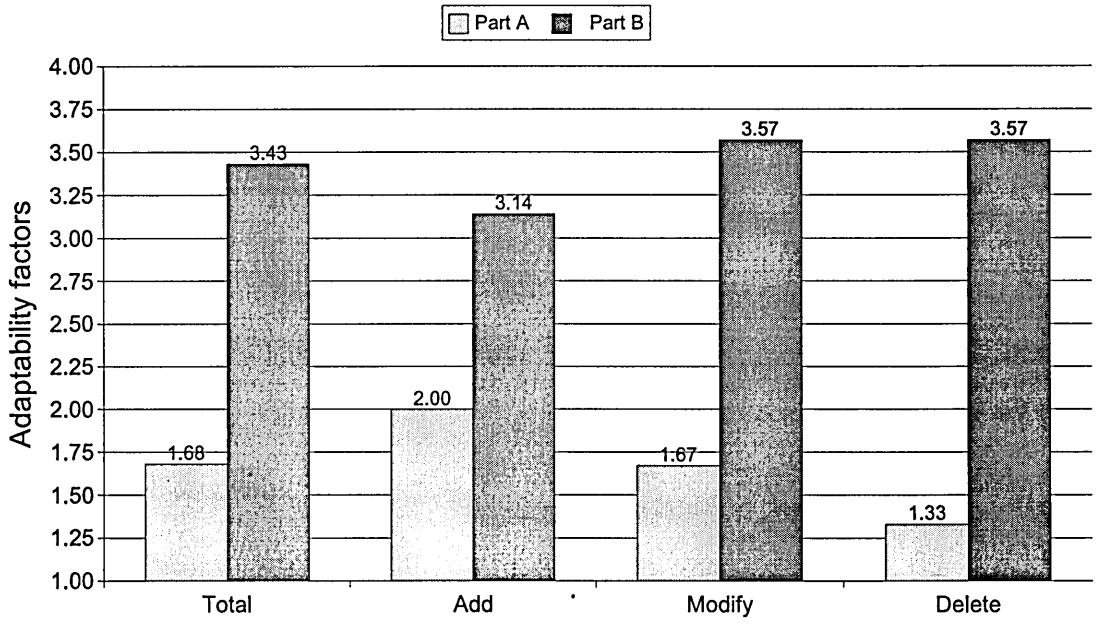

Figure 7.1: Adaptability factors

It can safely be assumed that, due to the immediate comparison of the two models with identical content, the modifier was led to rank the adaptability of the models more extremely than if each model was looked at on its own.

However, this ranking, showing the model built without the model building guidelines/adaptability guidelines on the difficult end of the scale, and the model, built using the guidelines introduced in chapter 6, on the other end of the scale, suggests that those guidelines do increase the overall adaptability of the model.

The comments made by the modifier, shown in full in Appendix F, reinforce the results shown above, stating repeatedly that version *B* was easier to understand and easier to modify.

Only the "modifier's" knowledge of, and tendency to use, VBA and externally linked data, resulted in her opinion that *option 3* from the data placement matrix (see section 6.3.4 and figure 7.2 on the following page) should have been used.

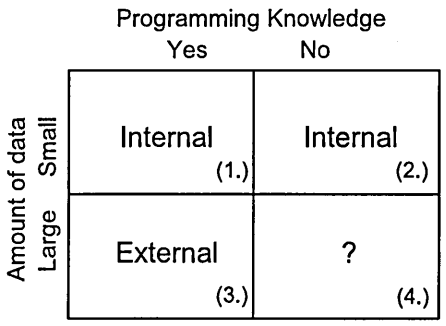

Figure 7.2: Data placement matrix

7.3 Key findings

7.3.1 Interviews The interview results suggest, at first glance, that the *model building guidelines* would be of little additional help in building simulation models. A lot of items listed in them seem superfluous, or repetitions of items from another part of the guidelines. The interview also showed, however, that some parts of the guidelines were quite unclear and vague.

This suggests that the problems associated with those guidelines come from insufficiently clear language and lack of examples.

As a result, the guidelines have been modified somewhat, and examples included, to improve understanding of those guidelines, before they were applied in the experiment.

7.3.2 Experiment The results obtained from the experiment, conducted using the updated, improved model building guidelines, put the version of the model built without guidelines on the difficult end of the scale, and the one built using the guidelines on the opposite end of the scale.

This very substantial difference in how the models were graded on their adaptability, showing that it was overall very difficult (average rating of 1.68) to modify the version built without guidelines, while giving a rating of 3.43 to the one built using the guidelines, strongly suggests that the guidelines proposed in chapter 6 do indeed substantially increase the adaptability of a model.

Chapter 8

Findings and Conclusion

In this chapter, the results obtained during the research and laid out in the preceding chapters, are looked at in more detail so as to deduce valid conclusions.

8.1 Interviews

The interviews were not so much a means to gather quantifiable data, but to gain an initial insight into the processes involved in building a simulation model and managing a simulation project. Those insights and the main points of critique of existing simulation modelling software were of paramount importance to construct the questionnaire survey.

The main outcome was that access to the simulation code was extremely important, while animation facilities where only of secondary importance. This secondary importance was due to the fact that, in a majority of cases, the results where presented to people with good knowledge of the domain simulated. This eliminated the need for complex, fancy animation, or rendered animation as a whole obsolete. However, it also emerged that animation must not be totally eliminated, as it is useful in the validation process. In the case of simulation with AutoMod, the facility to export a play-back animation (of an earlier simulation project) as a video file proved helpful to give customers an idea of what they can expect.

8.2 Questionnaire survey

8.2.1 User related findings While Hlupic (2000) survey focused solely on members of the '*Simulation Study Group of the Operational Research Society of Great Britain*', the

basis for this survey were simulation software users from all over the world. Nonetheless, Arena is used by a roughly a third of all users in both cases. This survey, however, concludes that Arena is the overall most widely used single simulation package. This top ranking, compared to the 3^{rd} place in Hlupic's survey, can be attributed to two possible causes (discounting a difference due to smaller sample size):

1 different usage patterns in terms of software packages used in the UK and elsewhere
2 the improvement of software packages, and, in particular, of Arena over the last 4 years (Hlupic refers to SIMAN/Cinema, the precursor of Arena, rather than to Arena itself)

As figure 4.1 on page 66 shows, 62 % of simulation software users have built at least 10 models (roughly 56 % of academic and 66 % of industrial users). While not in themselves very significant, these numbers provide greater validity and reliability to the results obtained in the main question bloc.

The ratio of 39 % academic to 61 % industrial users (based on exclusive academic and industrial users, with double entries not taken into consideration) proves that simulation is an important tool in industry. This, however, contradicts the findings by Hlupic (2000), who reported a ratio of 66 % academic versus 33 % industrial users. But, here again, it must be taken into account that Hlupic based her study solely on simulation use in the UK. As indicated by Murphy and Perera (2001a), the use of simulation is much more widespread in industry in the US.

8.2.2 Main findings As can be gathered from table 4.2 on page 69, deleting various elements (i.e. entities, resources and handling devices) from a simulation model is generally considered somewhat easier than modifying elements, while adding is the relatively most complicated task. More precisely, the task of adding an element (irrespective of whether it is an entity, a resource or a handling device) is considered *very easy* or *somewhat easy* by 75.1 % of respondents, modifying it is considered *very easy* or *somewhat easy* by 77.8 %, while 79.8 % of respondents replied that deletion was the easiest task.

The data also show that entity-related operations (i.e. adding, modifying and deleting) are easiest to perform and handling device-related ones are the most complex.

Similarly to the results obtained on adding, modifying and deleting elements (see above), removing model detail was considered marginally easier (66.28 % of respondents

ticked *very easy* or *somewhat easy*) than adding model detail, which 60.92 % of respondents considered *somewhat* or *very easy*.

Modifying the distribution (see table 4.4 on page 71), i.e. changing from, e.g., an exponential to a normal distribution, is *very easy* for 58 % of respondents, while 19 % thought of this as *somewhat* or *very difficult*. Changing the data form, on the other hand, is *somewhat* or *very difficult* for 42 % of respondents. (see table 4.4 on page 71) The difference between changing the distribution and changing the data form is understandable, as most modern simulation packages offer some sort of drop-down facility (or something similar) to quickly modify the distribution. Switching between types of data, however, is not as straight forward, and changing from e.g. stochastic to empirical data often necessitates access to external data files as well as some programming knowledge.

As table 4.6 on page 71 indicates, 40.91 % of respondents change the model logic frequently and 15.91 % always change it, once a working, valid model has been created, and only 3.41 % of simulation software users never modify the logic of a valid model.

Table 4.9 on page 72 shows that in roughly 43 % of cases, a change of simulation objectives is the reason for modifying the model logic. In 21 % of cases it is due to a major change of the underlying real-world system. In 22 % of all cases, where the logic is modified, is it due to a change to the required amount of detail and in 15 % of all cases, other reasons contribute to having to modify the logic of a validated, properly running model.

> "*...packages are hard to use well on problems that diverge even a little from what the designer had in mind, and off-the-shelf software means the answer to somebody else's problem*" (Hlupic 2000).

Nonetheless, it is interesting to note that among the 80 % of respondents who have templates available (see table 4.7 on page 71), 38 % use them *frequently*. 16 % of respondents *always* use them and the same amount *never* use templates (see table 4.8 on page 71). With more than half of respondents making use of available templates on a frequent to permanent basis, it seems that the templates developed by simulation software vendors are a very welcome addition.

Finally, the results pertaining to the difficulty in changing the animation (see table 4.10) reveal that 45 % of respondents consider changing an animation *somewhat easy* and 24 % as *somewhat difficult*. 26 % find such modifications *very easy* and only 4.5 %

very difficult. The fact that 71.6 % of respondents rated the difficulty as somewhat or very easy shows that the animation functions provided by the simulation packages are quite sufficient.

8.2.3 conclusion The survey data reviewed here reveal overall that deleting or removing elements or model detail is somewhat easier than adding or modifying it, that templates are being put to use and that the possibility to access the simulation code itself is very useful. However, it also shows that there is a lot of potential to simplify the main, basic tasks involved in creating or modifying simulation models as well as the animation.

From this emanates that any add-on module or facility to simplify those tasks would greatly help simulation software users. The next step to be taken should therefore consist of finding a solution to provide this extended flexibility and incorporate it into existing software packages.

8.3 Benchmarking

The benchmarking results show a number of things.

Firstly, they confirm the findings from the questionnaire survey that modifying is the most difficult task, while deleting elements, such as entities (products), resources or handling devices, is the easiest task.

Secondly, the benchmarking shows that sequencing does overall increase the adaptability, especially for adding and modifying elements. For deleting them, sequencing seems to offer no advantages. However, to the extent to which this was tested, the increase in adaptability was small for both packages tested. For both packages, the overall increase in adaptability by using sequencing amounted to 1%, and between 1% and 7% per category, with one exception. For the modification of elements in Simul8, the increase in adaptability when sequencing was used, compared to hard coding, amounted to 23%.

Thirdly, the results from the benchmarking exercise suggest that the Arena COTS simulation package is slightly better suited for adaptability than Simul8. However, while Arena required slightly fewer steps than Simul8 for a given modification, the author had the subjective feeling that Simul8 was overall easier to handle.

Finally, the benchmarking showed that the use of sub-models, or, by inference, customized templates, hinders changing components contained in them, rather than making changes easier to implement. Their use should therefore be limited if changes to such subcomponents, by another person than the original model builder, are likely.

8.4 Modified model building guidelines

Based on the results obtained from the survey and benchmarking (see chapters 4 and 5 respectively) as well as the literature on the subject (see chapter 2), it became evident that there was a lack of model building guidelines specifically designed to aid in future modification/reuse of a given model.

As a result, a modified model building guidelines, based on the *steps in a simulation study* devised by Law and Kelton (2000), were developed.

These new guidelines put strong emphasis on the way a model is structured and laid out, encourage a stronger, closer cooperation with higher management and stronger association with mid- to long-term company strategy. They also make certain parts of the documentation more explicit, while at the same time suggesting the use of easy to understand terms for e.g. variables and attributes, as much to make the documentation clearer as to help in modifying the model at a later stage.

This new approach to model building was then validated in an interview with a simulation practitioner with many years' practice. As a result of this interview, some modifications to the guidelines had to be made and some elements clarified.

Next, those updated guidelines were used in a framed field experiment, where identical simulation requirements where handed to two simulation practitioners. One of them was also given the guidelines.

The models these two practitioners built were then passed on to a third simulation practitioner, who then had to modify them, by implementing the same changes in both of them (ie based on a new set of simulation requirements), and rating how easy – or difficult – it was to implement those changes in each model.

From the results thus obtained, it clearly shows that the model built using the modification guidelines is substantially easier to modify.

8.5 Summary of conclusion

In section 1.3 on page 4, the following questions were 13put forward as the basis for this work:

i **What change or changes to a simulation project constitute the biggest problem, at its implementation, for simulation software users?**

ii **How can those changes be avoided?**

iii **How can the implementation of changes to a simulation project be simplified?**

These main questions were then worked on by means of addressing the following main issues:

i Investigation of current practices of building adaptable simulation models in industry

ii Identification of the most difficult and/or most frequent changes to implement in a simulation study, irrespective of the software used,

iii Identification of parameters which describe the adaptability of simulation models

iv Conducting a benchmarking study of some of the leading simulation software packages towards their inbuilt adaptability

v Development model building guidelines for the creation of adaptable simulation models

vi Development of a prototype adaptable simulation model and validation of the design guidelines.

The investigation of current simulation practices was done as a literature review, mainly centred around simulation guidelines and the current status of adaptable simulation modelling, including model reuse and adaptation.

The interviews and questionnaires were used to identify the most frequent and most difficult changes to simulation models. From those emanates that modifications of the code in whatever form (e.g. via typing code or rearranging graphical components) outweigh the frequency at which animations are modified, and also outranks the importance of modifying animations, as that is only secondary to the simulation and its output data.

The most difficult changes, according to the user survey, were the ones involving a change to the form of data (e.g. from internal to external data). Table 8.1 ranks the different modifications with increasing ease of implementation. The percentages relating to *elements* are based on table 4.2 on page 69, those relating to modifying distributions

and data forms on table 4.5 on page 71, while the ratings for adding and removing model detail are based on table 4.6 on page 71.

Table 8.1: Overall difficulty rating

Rank	Action	Ease
1	Change data form	58.52%
2	Add model detail	60.92%
3	Remove model detail	66.28%
4	Add elements	75.10%
5	Modify elements	77.78%
6	Delete elements	79.78%
7	Change distribution	80.23%

Rating based on *very easy* and *easy* categories

The identification of parameters describing the adaptability of simulation models led, on the one hand, to the classification categories of:

1 data location adaptability
2 data type adaptability
3 logic adaptability
4 animation adaptability
5 interface adaptability

(see section 5.1 on page 74 for details), and, on the other hand, to a slight modification of the equation by Herrmann et al. (2000), with the modified version shown in equation 8.5.1

$$A_{(M_0,M_1)} = 1 - \frac{E_{(M_1-M_0)}}{E_{(M_1)}} \tag{8.5.1}$$

Those adaptability parameters, and the modified equation, were then used to benchmark two of the most widely used simulation packages, i.e. Arena and Simul8. The benchmarking was conducted by performing a list of changes, including the addition, modification and deletion of elements (entities), resources, handling devices, attributes etc. From this benchmarking, it emanated that Arena was overall slightly more adaptable, even though Simul8 *felt* easier to handle, and that the use of sequencing, overall, does somewhat improve the adaptability.

The next step in this research was to develop novel design guidelines for adaptable simulation models.

Those model building guidelines, described in great detail in chapter 6 and in Appendix G, were developed mainly for somewhat irregular/infrequent simulation package

users, and specifically with the aim of making the modification of an existing simulation model at some point in the future easier.

They where then validated by means of an interview, the feedback of which was then used to improve the guidelines. Next, the updated guidelines were submitted to a framed field experiment, to gauge their effectiveness against *conventional* model building practices, i.e. practices whose aim it is not to improve future modification of an existing simulation model.

While the interview suggested that the guidelines were most likely of little help, the experiment showed that their use during the building of a model does substantially increase the ease of implementing modifications later on.

8.6 Limitations and further work

The first stage of validating the modified methodology consisted of an interview with only one simulation practitioner. While this is clearly insufficient for establishing any form of statistical reliability, it was still very helpful.
However, the interviewee's extensive knowledge of various simulation packages and his substantial experience as simulation consultant, were of very reliable and high value to the author, and for the validation.

The framed field experiment, consisting of one experimental model and one control model, both of which were modified by the same person, also contains some limitations. While the results obtained exceeded the author's expectations, the fact that only one such experiment could be conducted clearly constitutes a caveat.

In order to improve on these issues, further work could include repeating the framed field experiment with other groups of people.

Furthermore, the model building guidelines presented in this thesis could be tested against non-simulators, eg simulation software based on special-purpose programming languages without graphical interfaces, to test the validity of the guidelines beyond their original use of COTS-packages.

Also, as the above mentioned guidelines are kept quite general, i.e. applicable to a wide range of COTS-packages, future work in this area could entail the creation of

package-specific modelling guidelines, as such simulator-specific guidelines might further improve the adaptability of a model built using them in that specific package.

References

ARENA06. Rockwell – arena simulation, 2006. `http://www.arenasimulation.com/news/default.asp`.

O Balci. Guidelines for successful simulation studies. In O Balci, RP Sadowski, and RE Nance, editors, *Proceedings of the 1990 Winter Simulation Conference*, pages 25–32, December 1990.

LP Baldwin, T Eldabi, V Hlupic, and Z Irani. Enhancing simulation software for use in manufacturing. *Logistics Information Management*, 13(2):263–270, 2000.

J Banks. The future of simulation software: A panel discussion. In A Andradóttir, KJ Healy, DH Withers, and BL Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, pages 166–173, December 1997.

J Banks. Simulation in the future. In JA Joines, RR Barton, K Kang, and PA Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 1568–1576, December 2000.

J Banks and JS Carson. *Discrete-event system simulation*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984. ISBN 0132155826.

P Benjamin, D Delen, R Mayer, and T O'Brien. A model-based approach for component simulation development. In JA Joines, RR Barton, K Kang, and PA Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 1831–1839, December 2000.

L Blaxter, C Hughes, and M Tight. *How to research. Second Edition*. Open University Press, Buckingham, 2001. ISBN 0335209033.

A Bryman. *Quantity and Quality in Social Research*. Routledge, London, 1988. ISBN 0415078989.

JA Buzacott. The fundamental principles of flexibility in manufacturing systems. In *Proceedings of the First International Conference in Flexible Manufacturing Systems*, pages 13–21, North Holland, Amsterdam, October 1982. Elsevier Science Publishers B.V.

RE Callan. *Building Object-oriented Systems: An introduction from concepts to implementation in C++*. Computational Mechanics Publications, Southampton, 1994. ISBN 1853123404.

F Chance, J Robinson, and J Fowler. Supporting manufacturing with simulation: model design, development, and deployment. In JM Charnes, DJ Morrice, DT Brunner, and JJ Swain, editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 114–121, December 1996.

WJ Davis and GL Moeller. The high level architecture: Is there a better way? In PA Farrington, HB Nembhard, DT Sturrock, and GW Evans, editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 1595–1601, December 1999.

KG de Ruiter, JM Sluijs, and WB Stoutjesdijk. Simulation for recurring decisions. In JA Joines, RR Barton, K Kang, and PA Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 1217–1221, December 2000.

G de Vreede, A Verbraeck, and DT van Eijck. Integrating the conceptualization and simulation of business processes: A modeling method and arena template. *Simulation*, 79(1):43–55, January 2003.

T Eldabi and RJ Paul. Flexible modeling of manufacturing systems with variable levels of detail. In S Andradóttir, KJ Healy, DH Withers, and BL Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, pages 801–808, December 1997.

T Eldabi and RJ Paul. Evaluation of tools for modeling manufacturing systems design with multiple levels of detail. *The International Journal of Flexible Manufacturing Systems*, 13(2):163–176, 2001.

SM Gahagan and JW Herrmann. Improving simulation model adaptability with a production control framework. In BA Peters, JS Smith, DJ Medeiros, and MW Rohrer, editors, *Proceedings of the 2001 Winter Simulation Conference*, pages 937–945, December 2001.

D Garlan, R Allen, and J Ockerbloom. Architectural mismatch or why it's hard to build systems out of existing parts. In *Proceedings of the Seventeenth International Conference on Software Engineering*, Seattle, WA, June 1995.

SI Gass. Computer model documentation. In *WSC '78: Proceedings of the 10th conference on Winter simulation*, pages 281–287, Piscataway, NJ, USA, 1978. IEEE Press.

D Gerwin. An agenda for research on the flexibility of manufacturing processes. *International Journal of Operations and Production Management*, 7(2):38–49, January 1987.

D Gerwin and H Kolodny. *Management of Advanced Manufacturing Technology: Strategy, Organization and Innovation*. John Wiley & Sons, Inc., New York, 1992. ISBN 047163574X.

RL Gorden. *Interviewing - Strategy, techniques and tactics. Revised ed.* The Dorsey Press, London, 1975. ISBN 0256015112.

GW Harrison and JA List. Field experiments. *Journal of Economic Literature*, 42:1009–1055, 2004.

AJ Head. *Design Wise: A guide for evaluating the interface design of information resources*. CyberAge Books, Medford, N.J., 1999. ISBN 0910965315.

JW Herrmann, E Lin, B Ram, and S Sarin. Adaptable simulation models for manufacturing. In *Proceedings of the 10th International Conference on Flexible Automation and Intelligent Manufacturing*, pages 989–995, College Park, MD., 2000. University of Maryland, Department of Mechanical Engineering. Volume 2.

V Hlupic. Simulation software: An operational research society survey of academic and industrial users. In JA Joines, RR Barton, K Kang, and PA Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, pages 1676–1683, December 2000.

ISBSG05. International software benchmarking standards group, 2005. `http://www.isbsg.org/`.

JA Joines and SD Roberts. Fundamentals of object-oriented simulation. In DJ Medeiros, EF Watson, JS Carson, and MS Manivannan, editors, *Proceedings of the 1998 Winter Simulation Conference*, pages 141–150, December 1998.

RA Jones and JM Ostroy. Flexibility and uncertainty. *Review of Economic Studies*, 51 (164):13–32, January 1984.

DM Keats. *Interviewing : a practical guide for students and professionals*. Open University Press, Buckingham, 2000. ISBN 0335206670.

WD Kelton, RP Sadowski, and DA Sadowski. *Simulation with Arena. 2nd ed.* McGraw-Hill, New York, 2002. ISBN 0072392703.

A Ku. *Modelling Uncertainty in Electricity Capacity Planning*. PhD thesis, London Business School, 1995.
`http://www.analyticalq.com/thesis/default.htm`.

BC Kulick and JT Sawyer. A flexible interface and architecture for container and intermodal freight simulations. In PA Farrington, HB Nembhard, DT STurrock, and GW Ewans, editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 1238–1242, December 1999.

AM Law and WD Kelton. *Simulation modeling and analysis. 3rd ed.* McGraw-Hill, Singapore, 2000. ISBN 0071165371.

YT Lee, C McLean, and G Shao. A neutral information model for simulating machine shop operations. In S Chick, PJ Sánchez, D Ferring, and DJ Morrice, editors, *Proceedings of the 2003 Winter Simulation Conference*, pages 1296–1304, December 2003.

M Mandelbaum. *Flexibility in Decision-Making: An Exploration and Unification*. PhD thesis, Department of Industrial Engineering, University of Toronto, 1978.

C McLean, A Jones, YT Lee, and R Riddick. An architecture for a generic data-driven machine shop simulator. In E Yücesan, CH Chen, JL Snowdon, and JM Charnes, editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 1108–1116, December 2002.

C McLean and S Leong. The role of simulation in strategic manufacturing. In *Proceedings of the Fourth International Workshop of the IFIP WG 5.7 Special Interest Group (SIG) on "Advanced Techniques in Production Planning & Control"*, Aalborg, Denmark, August 2001. Special Sessions of the IFIP WG 5.7 International Working Conference on Strategic Manufacturing.

C McLean and S Leong. A framework for standard modular simulation. In E Yücesan, CH Chen, JL Snowdon, and JM Charnes, editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 1613–1620, December 2002.

SP Murphy and TD Perera. Key enablers in the development of simulation. In BA Peters, JS Smith, DJ Medeiros, and MW Rohrer, editors, *Proceedings of the 2001 Winter Simulation Conference*, pages 1429–1437, December 2001a.

SP Murphy and TD Perera. Successes and failures in uk/us development of simulation. *Simulation Practice and Theory*, 9(6-8):333–348, 2001b.

OED93. *The New Shorter Oxford English Dictionary*, Oxford, 1993. Oxford University Press. ISBN 0198612710.

J Oscarsson and M Urenda Moris. Documentation of discrete event simulation models for manufacturing system life cycle simulation. In E Yücesan, CH Chen, JL Snowdon, and JM Charnes, editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 1073–1078, December 2002.

SP Parker. *McGraw-Hill Dictionary of scientific and technical terms. 4th ed.* McGraw-Hill, New York, 1989. ISBN 0070452709.

M Pidd. Guidelines for the design of data driven generic simulators for specific domains. *Simulation*, 59(4):237–243, 1992.

M Pidd. Simulation software and model reuse: A polemic. In E Yücesan, CH Chen, JL Snowdon, and JM Charnes, editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 772–775, December 2002.

M Pidd, N Oses, and RJ Brooks. Component-based simulation on the web? In PA Farrington, HB Nembhard, DT Sturrock, and GW Evans, editors, *Proceedings of the 1999 Winter Simulation Conference*, pages 1438–1444, December 1999.

T. Pohl. Questionnaire survey. Technical report, Sheffield Hallam University, Sheffield, UK, 2003. Internal Working Document.

PSBS05. Public sector benchmarking service (psbs), 2005. `http://www.benchmarking.gov.uk/`.

SJ Ravden and GI Johnson. *Evaluating Usability of Human-Computer Interfaces: A Practial Method.* Ellis Horwood Limited, Chichester, 1989. ISBN 0745806147.

R Reese and DL Wyatt. Software reuse and simulation. In A Thesen, H Grant, and WD Kelton, editors, *Proceedings of the 1987 Winter Simulation Conference*, pages 185–192, December 1987.

N Robertson and TD Perera. Automated data collection for simulation? *Simulation Practice and Theory*, 9(6-8):349–364, 2001.

S Robinson. *Successful simulation – A Practical Approach to Simulation Projects.* McGraw-Hill, Berkshire, 1994. ISBN 0077076222.

salfi06. Field experiments, 2006. http://www.chssc.salford.ac.uk/healthSci/resmeth2000/resmeth/field.htm.

N Slack. Flexibility as a manufacturing objective. *International Journal of Operations and Production Management*, 3(3):4–13, 1983.

N Slack. *The manufacturing advantage: achieving competitive manufacturing operations.* Management Books 2000 Ltd, Cirencester, 1991. ISBN 1852510382.

YJ Son, AT Jones, and RA Wysk. Component based simulation modeling from neutral component libraries. *Computers & Industrial Engineering*, 45(1):141–165, June 2003.

M Spiegel, PF Reynolds Jr., and DC Brogan. A case study of model context for simulation composability and reusability. In ME Kuhl, NM Steiger, FB Armstrong, and JA Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 437–444, December 2005.

JS Swain. Simulation reloaded: Sixth biennial survey of discrete-even software tools. *OR/MS Today*, 30(4):46–57, August 2003.

BS Tye. *A Study on Model Design in the Simulation of Manufacturing Systems.* PhD thesis, Sheffield Hallam University, 1999.

KE Weick, LB Barnes, T Burns, and TB Roby. *Methods of Organizational Research.* University of Pittsburgh Press, Pittsburgh, Pennsylvania, 1968. Edited by VH Vroom.

M Weihua and C Baofeng. Review on manufacturing flexibility: Definition and classification. In *Proceedings of the 99 International Conference on Agricultural Engineering*, pages 131–134, Beijing, China, 1999. China Agricultural University Press.

wikiex06. Experiments, 2006. `http://en.wikipedia.org/wiki/Experiment`.

L Yilmaz and T Ören. A conceptual model for reusable simulations within a model-simulator-context framework. In *Proceedings of CMS 2004 – Conference on Conceptual Modeling and Simulation*, pages 235–241, Genoa, Italy, October 2004.

RK Yin. *Case Study Research – Design and Methods, Third ed.* Sage Publications Ltd, London, 2003. ISBN 0761925538.

W Zhao and A Verbraeck. A framework for configurable hierarchical simulation in a multiple-user decision support environment. In ME Kuhl, NM Steiger, FB Armstrong, and JA Jones, editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 327–335, December 2005.

Appendix A

Interview Questions

Project Duration

- min project duration (weeks or months?)
  type of project (i.e. recurring type?) why so short?
- max project duration (weeks or months?)
  why so long?
- estimated average project duration (weeks or months?)
  type of project (i.e. recurring type?)

Model Building Methodology

Which of the included models most closely resembles the approach you take?

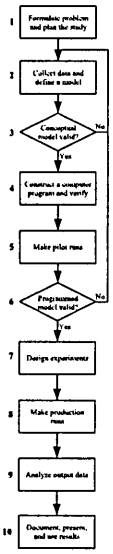

Figure 1: Steps in a simulation study (Law & Kelton, 2000, p. 84)

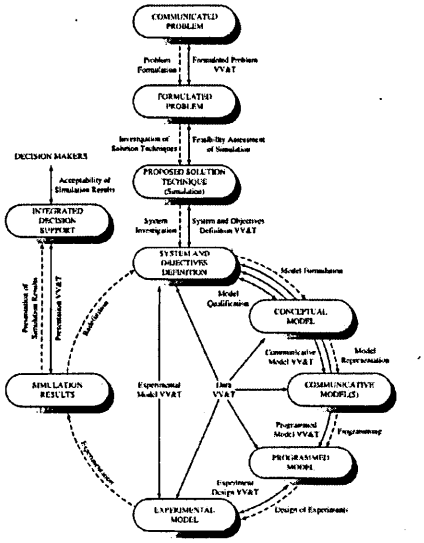

Figure 2: The life cycle of a simulation study (Balci, 1990, p. 26)

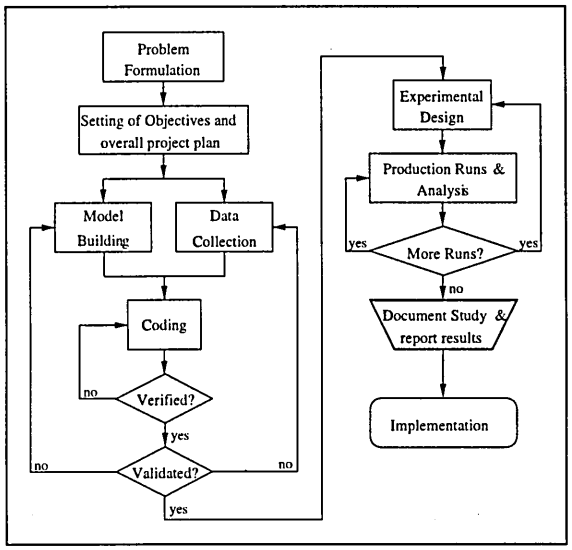

Figure 3: Steps in a simulation study (Banks, Carson, 1984, p. 12)

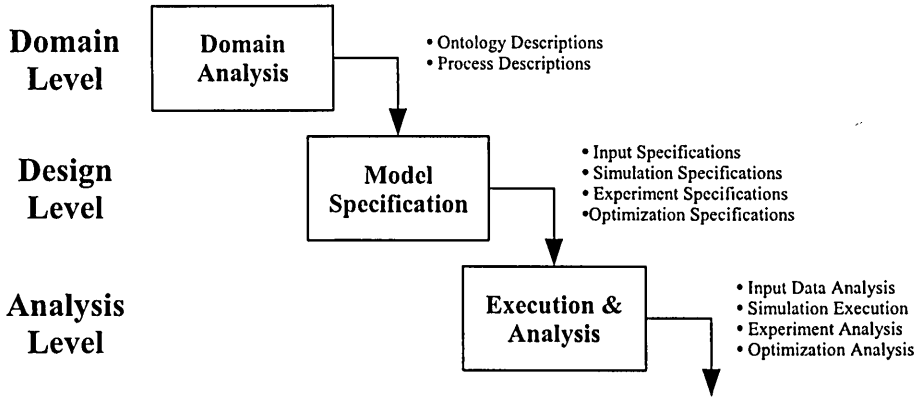

Figure 4: Separation of Levels (Benjamin et al, 2000, p. 1832)

Types and frequency of changes

- What type of changes occur?
  - (eg objectives, logic, type or amount of data etc)
  - which category leads to more problems, which is easier/more difficult to implement
  - what/who influences those changes?
  - recurrence of changes (frequency of changes, frequency of most difficult to implement)
- most frequent (1) to least frequent (5)
  - objectives
  - logic
  - type of data (internal vs. external, empirical vs. stochastic, different distributions)
  - number of elements (entities, resources)
  - type of experiments (alternative set-ups)

Appendix B

Questionnaire Survey

Design of Flexible Simulation Models

As part of a three year research project studying the flexibility of simulation models, we are investigating the features of the simulation software used by academia and industry. We would be grateful if you could spend a few moments filling in this short questionnaire.

If you would like further information on our work, please contact us at the School of Engineering Sheffield Hallam University, England.

Tom Pohl - Thomas.Pohl@student.shu.ac.uk **OR** Dr Terrence Perera - T.D.Perera@shu.ac.uk

This questionnaire can also be filled in online at
http://vi05n108.members.eunet.at/survey/survey.html

What type of simulation software user are you?

[ ] Academic [ ] Industrial [ ] Government

[ ] Other (Please specify) .....

What is your current profession?

(Please specify) .....

How many simulation models have you built so far?

[ ] < 4 [ ] 4 < 7 [ ] 7 < 10

[ ] 10 < 20 [ ] 20 < 30 [ ] ≥ 30

Which simulation software package(s) do you primarily use?

(Please specify) ....

Please rank each of the following modifications, bearing in mind all the ensuing modifications to the model logic, excluding any animation (4 = easiest; 1 = hardest)

In case of modifying a model, e.g. deleting a resource, ensuing modifications may consist of having to change data relating to this resource in more than one place.

adding

- entities (e.g. product lines, raw material) [ ]
- resources (e.g. machines) [ ]
- material handling devices (humans, AGVs, conveyors...) [ ]

modifying

- entities (e.g. product lines, raw material) [ ]
- resources (e.g. machines) [ ]
- material handling devices (humans, AGVs, conveyors...) [ ]

deleting

- entities (e.g. product lines, raw material) [ ]
- resources (e.g. machines) [ ]
- material handling devices (humans, AGVs, conveyors...) [ ]

changing input data

- modifying the distribution (e.g. triangular to exponential) [ ]
- changing data form (e.g. distribution to data file) [ ]

adding model detail [ ]

removing model detail [ ]

How often do you modify a simulation model in order to reflect a change to the underlying system?

[ ] never [ ] sometimes [ ] frequently [ ] always

Does your simulation software package offer some form of generic model components (e.g. templates)?

[ ] yes [ ] no

If so, how often do you use them?

[ ] never [ ] sometimes [ ] frequently [ ] always

What is the main reason for reconfiguration of the model?

[ ] a major change to the underlying system

[ ] a change in the amount of detail required

[ ] changing simulation objectives

[ ] other (please specify) .....

When implementing a change (e.g. adding, deleting or modifying an element) in the animation, do you find this

[ ] very easy [ ] somewhat easy [ ] somewhat difficult [ ] very difficult

Does your software offer

an undo option [ ] yes [ ] no

a redo option [ ] yes [ ] no

Do you modify the simulation program's code in order to adapt a model to a new situation?

[ ] never [ ] sometimes [ ] frequently [ ] always

Do you see a limitation in the simulation software package you are using that makes modifying a simulation model particularly time-consuming or complicated to achieve?

.....

.....

.....

.....

.....

.....

Your e-mail address (***optional*** - only fill this in if you would be interested in participating in a potential follow-up questionnaire)

.....@.....

Appendix C

Organizational Layout

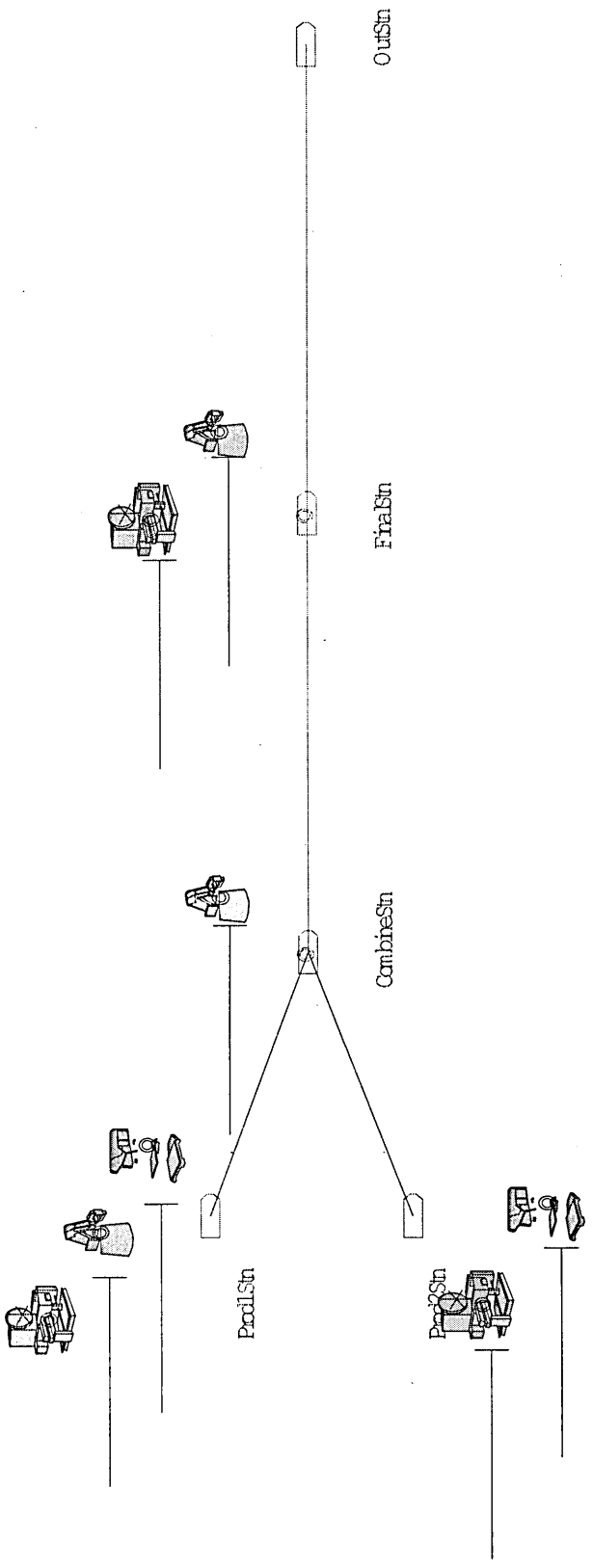

Figure 1: Animation

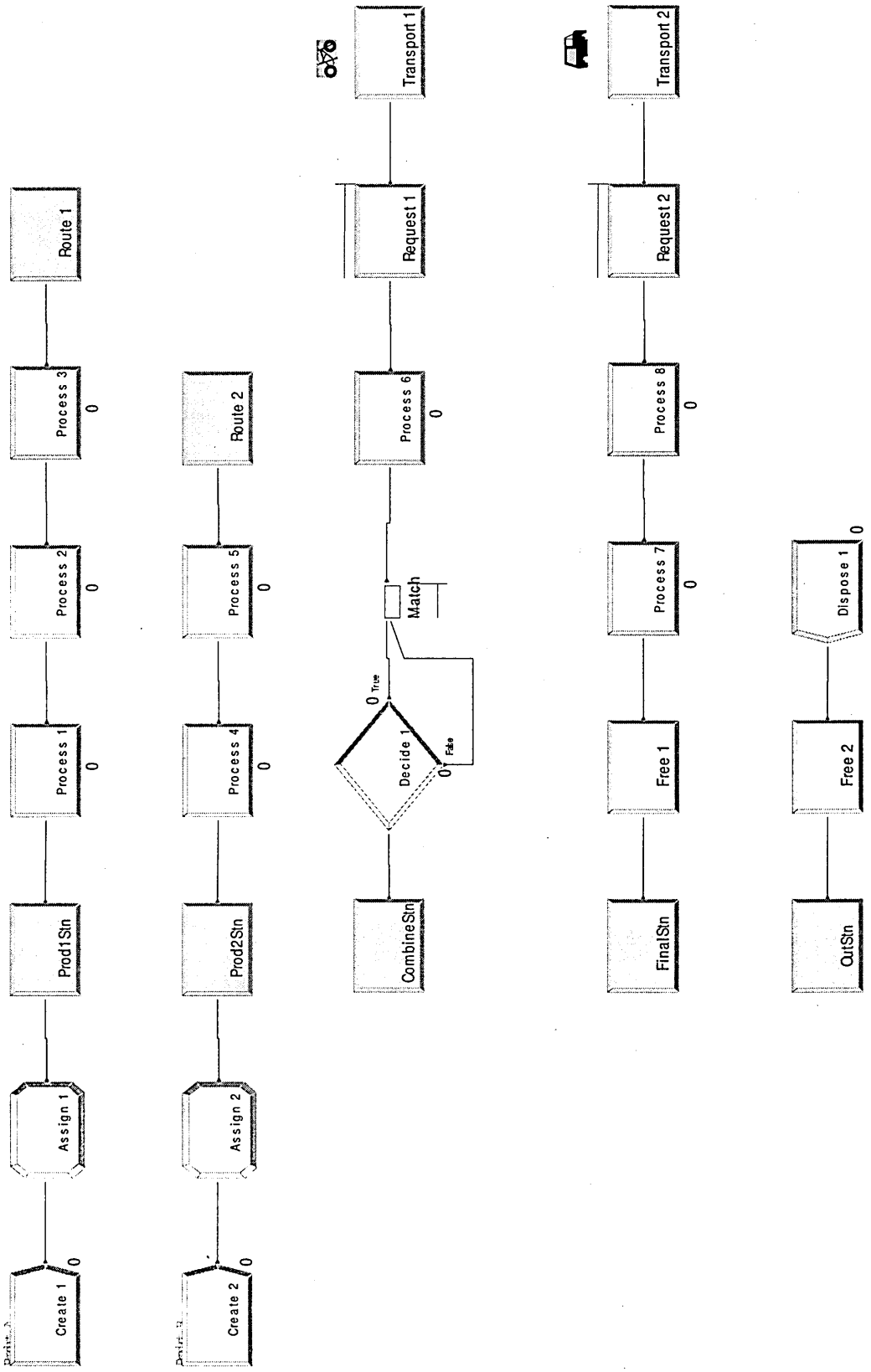

Figure 2: Simple logic

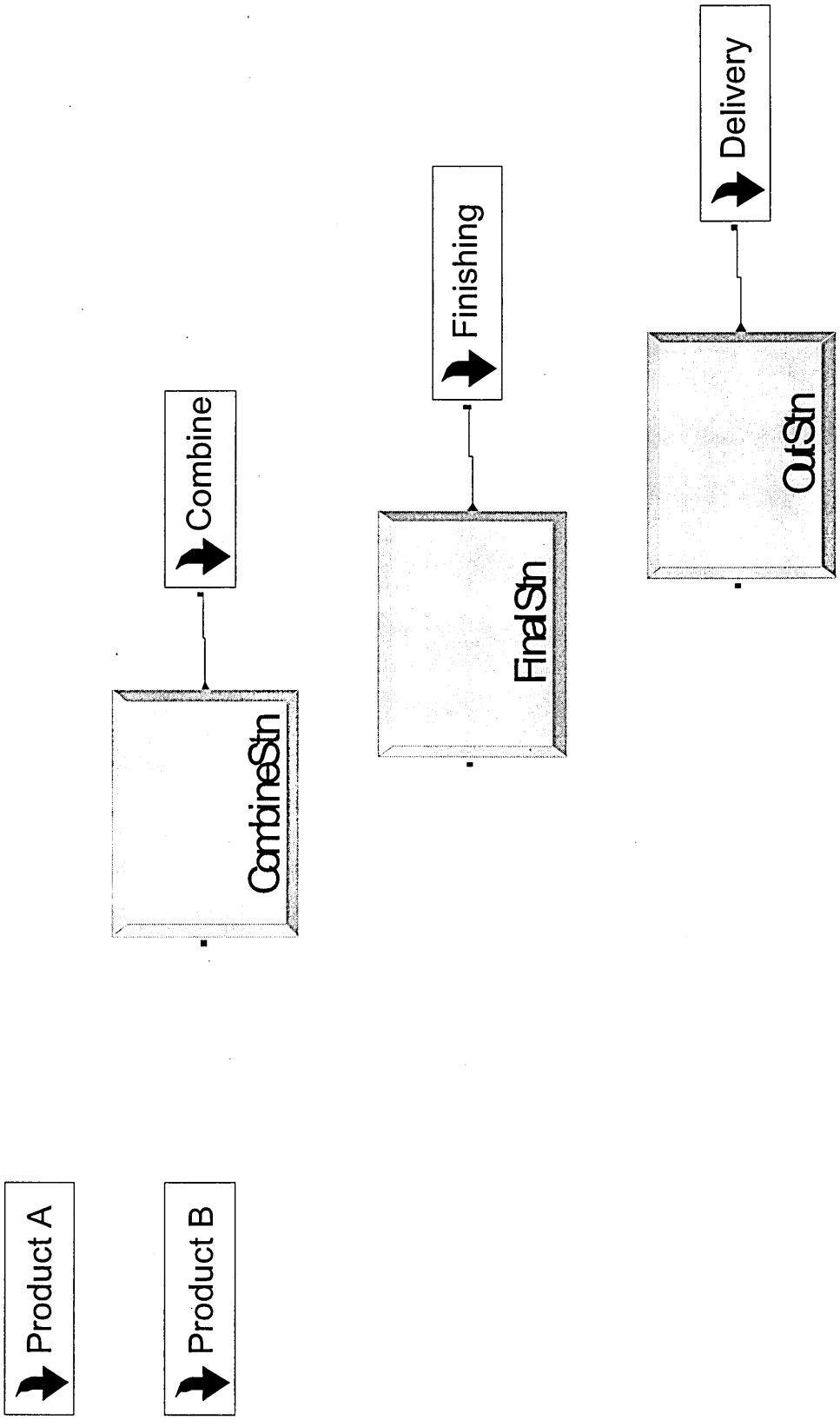

Figure 3: Submodel logic

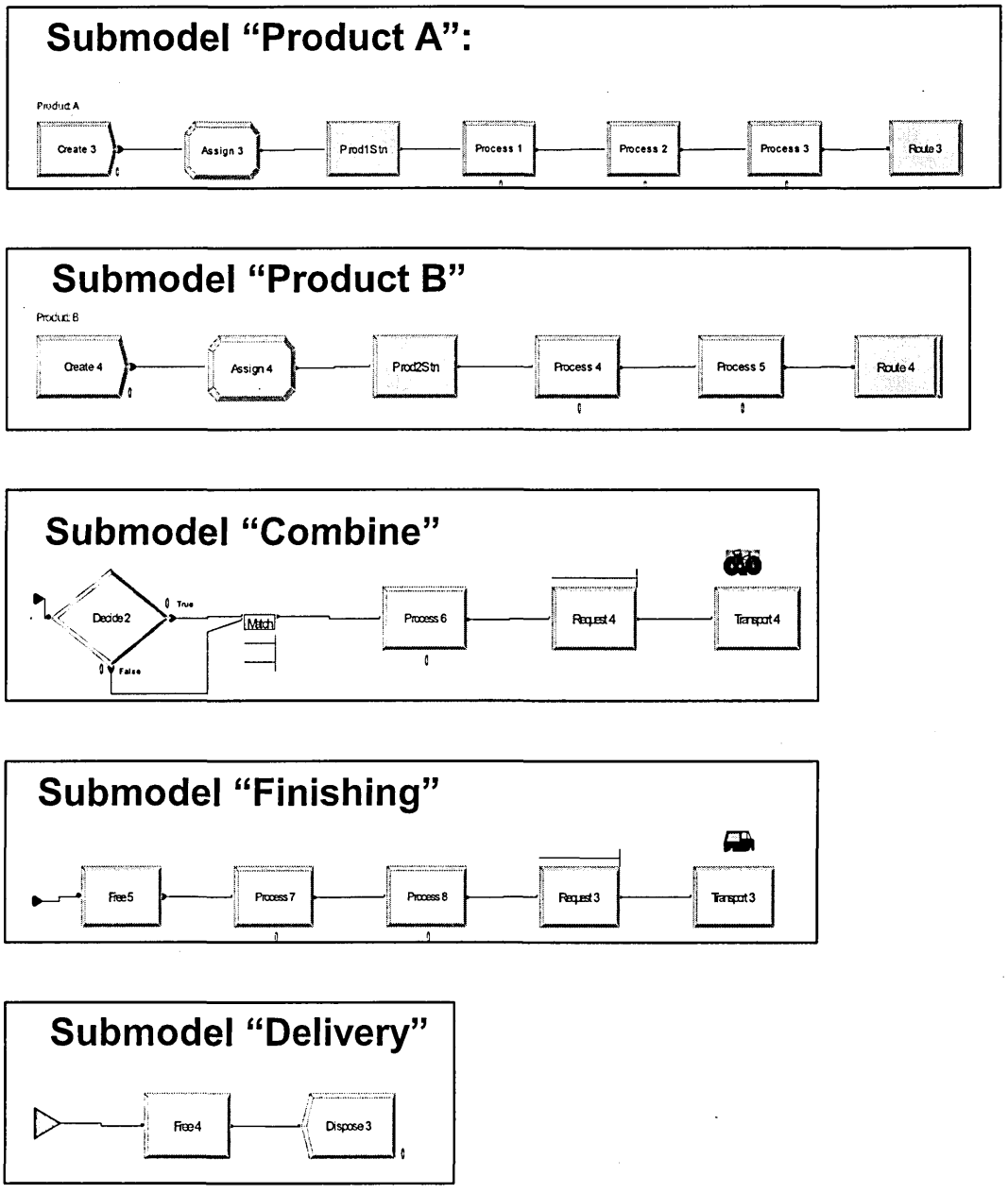

Figure 4: Submodel contents

Appendix D

Group Benchmarking Tasks

Sheffield Hallam University
Email: Thomas.Pohl@student.shu.ac.uk

Background

The former simulationist has left the company. He had built a simulation of the production plant for both products made at the company.

Since he left, management has decided to think about streamlining the existing production and introducing a new product.

You are the new simulationist, and have been asked to implement a few changes to the model your predecessor had built. (See ***Tasks***)

What you will need to know for these modifications

- Transporters and distances
- Conveyors and segments
- Attributes
- Processes (seize – delay – release)
- Decide blocs (choose/chance)

Tasks

1. Delete the third process from Product A (in Station Prod1Stn)
2. Create a third product, Product C, with interarrival times of EXPO(4) and Process Times of
   a. Process 6: NORM(2,0.06)
   b. Process 10: EXPO(3.7)
3. Create an additional station for product C, FinalCStn with process 10 (seize-delay-release – create a new resource Res10machine) and routing (via conveyor) to Delivery (i.e. OutStn) – see task 4 for conveyor
4. Include an additional transporter to transport elements from CombineStn to FinalCStn (distance: 20; load/unload times insignificant)
5. Include an additional conveyor to transport elements from FinalCStn to OutStn (distance: 20; load/unload times insignificant)
6. Increase the capacity of Resource 6 (Res6Machine) from 1 to 2
7. Exchange transporter 2 for a conveyor (distance remains the same, load and unload times are insignificant, i.e. do not need to be modelled)

Caution

- Due to the use of various handling devices, you may need to include additional "choose" blocs
- Although product C goes through the "Combine" station, it is NOT combined with any other product – this will require modification of the existing choose bloc
- Do not forget to create an additional sequence for Product C
- Do not forget to modify the animation accordingly
- IF, and only if, your model contains submodels, please create (a) new submodel(s) for Product C
- Where no specific data are given, enter what you think is adequate

Sheffield Hallam University
Email: Thomas.Pohl@student.shu.ac.uk

Please rate the following tasks after having implemented the changes in the Arena model:

		Very easy	Somewhat easy	Somewhat difficult	Very difficult
ADD	Product (entity)				
	Process (or Work Centre)				
	Attributes				
	Handling devices				
	Submodels*				
	Sequence				
MODIFY	Product (entity)				
	Process (or Work Centre)				
	Attributes				
	Handling devices				
	Submodels*				
	Sequence				
DELETE	Product (entity)				
	Process (or Work Centre)				
	Attributes				
	Handling devices				
	Submodels*				
	Sequence				

* *where applicable*

How helpful was the structuring of the Arena model you were given:
(Please tick as appropriate)

Very	*Somewhat*	*Hardly*	*Not at all*

Did you use submodels? (Please tick as appropriate)

Yes	*No*

Your experience with Arena: (Please tick as appropriate)

Poor	*Fair*	*Good*	*Very good*

Additional comments:

(continue on reverse if necessary)

Appendix E

Validation Interview Questions

User background

- Years experience
- Software(s)
- Number of projects (per software)
- Duration of projects

New modelbuilding guidelines

- What is your general opinion about those guidelines?
- Is graph easy to understand?
- Is the anticipation of future potential modifications a good idea?
- What do you think about the inclusion of higher management/long term strategy?
- Do you believe your COTS has sufficient internal annotation options?
- Is the preparation for future reuse by someone else helpful?
- Is the documentation of owners and keepers of further data and responsibility thereof good idea?
- Is the documentation of anticipated modifications (i.e. company strategy) good idea?
- Is there anything missing, incomplete or superfluous?

Adaptability guidelines

- What is your general opinion about the adaptability guidelines?
- Sequencing, submodels, annotation, data type (Your opinion)
- Is there anything missing, incomplete or superfluous?

(See also attached modelbuilding guidelines)

Validation Simulation – Part A

Tom Pohl
Sheffield Hallam University

August 16, 2006

1 Introduction

ITC (International Truck Corporation) produces 2 types of dump trucks, large and small ones, and in each case, there is a choice between trucks with and without liners. — The liners offer better protection against wear and tear when certain abrasive materials are transported in ITC vehicles.

Management has decided to have a simulation model of their truck floor production line built, to evaluate the optimum number of operators, and to see whether there is a potential bottleneck somewhere.

2 System Description

2.1 Layout and Processes

The layout in figure 1 on the following page shows the truck floor production line. The individual parts are delivered from the left into the respective parts buffers via dedicated transporters.

Depending on current orders, crane 1 or crane 2 picks up the required parts and delivers them to either the large or the small assembly station, where they are manually prepared for machine welding.

Once the truck floor is set up for welding, the assembled parts are transferred to the corresponding weld station via crane 3, where the weld robot then processes them.

After welding, crane 3 then picks up the welded truck floor and delivers it either to one of 5 output buffers or to the liner fitting station.

Once the truck floor has had the liner plates affixed – if so requested, it is either moved out of the production line with crane 4, or transferred to one of the output buffers, from which crane 4 then transfers them onto a dedicated transporter to move them on to the final truck assembly.

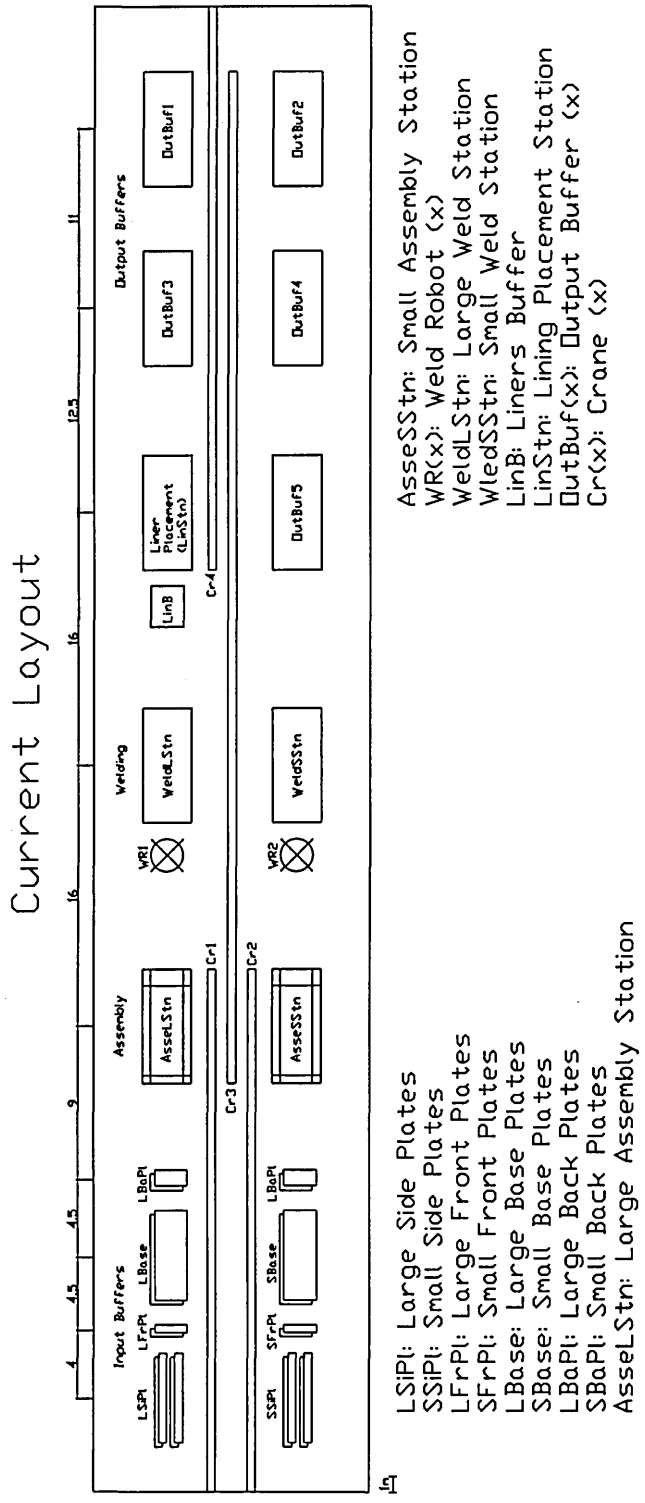

Figure 1: Current layout

2.2 Assumptions

2.2.1 Buffers

All buffers (base plates, front plates, back plates, side plates and liner plates) are regularly restocked from outside and do not run below a critical level.

As they are restocked from outside, by employees from a different part of the factory, this activity has no influence on the workload of the assembly line workers.

2.2.2 Cranes

In order to get the parts from one end of the production line to the other, the cranes have to overlap to some extent. However, they are all placed at different heights and do therefore not obstruct each other.

The cranes are operated by the assembly line workers who need them, e.g. by the assembly operators to put the parts together, by the welders to pick the assembled truck floor up for welding etc. I.e., all assembly workers are trained in crane operation.

There are two cranes available to assembly (cranes 1 and 2), three for the transfer between the assembly and welding (only one, crane 3, is shown in the diagram) and one crane to unload the liner fitting station and take care of the output buffers.

2.2.3 Assembly/Pre-weld

At the assembly stage, a worker spot-welds the parts together, which are held in place by crane 1 or crane 2 during spot welding.

Therefore, a worker can only either get a part for assembly, or spot-weld parts together.

2.2.4 Welding

While the majority of welding is done by the weld robot, some rework needs to be done manually, and a worker has to be present to monitor the welding process.

While there is one robot dedicated to large truck floors and one dedicated to small ones, the weld robots themselves are identical and only run slightly different programs. Changing the set-up from welding small truck floors to welding large ones, or vice-versa, takes 5min $\pm$10%.

2.2.5 Liner plate fitting

The placement of liner plates is done manually, helped by a small, stationary crane (not shown in figure 1 on the previous page).

2.2.6 Output buffers

There are currently 5 output buffers, and unloading those takes priority over unloading a weld or liner placement station, as the arrival of transporters to pick up finished truck

floors is determined by outside factors, and is sometimes a little sporadic. On average, a truck floor spends 5 hours±10% in the buffers.

3 Data

Orders for the assembly of truck floors are received roughly every 140 minutes (exponential distribution).

The current ratio of orders for small versus large trucks is 50% to 50%, with 30% of small trucks and 10% of large trucks requiring the addition of liner plates.

Table 1 shows the loading, unloading and process times. The loading/unloading times contain the cranes' travel time. The cranes' maximum speed under load is $0.5ms^{-1}$, empty $1.5ms^{-1}$.

Table 1: Load, unload and process times (in minutes)

	Small			Large		
Load time assembly	20	±	20%	20	±	20%
Assembly	98	±	5%	100	±	5%
Unload time (generic)	5	±	20%	5	±	20%
Load time (generic)	2	±	20%	2	±	20%
Weld time	46	±	2.5%	48	±	2.5%
Liner fitting time	89	±	2.5%	112	±	2.5%

The employees at ITC work in 3 shifts from Monday to Friday, although the Friday shifts are different from the rest of the week. Tables 3 on the next page and 4 on the following page show the specific shift patterns.

Per shift, there is one worker at each of the assembly stations, one at each weld station, one at the liner plate fitting station, and one who handles the unloading of the weld/liner fitting stations, loading and unloading of the output buffers. The operators have an efficiency of 80%.

Table 2 gives an overview of breakdown patterns for weld robots and manipulators. The other resources are assumed to be of so good a maintenance that breakdowns for them do not need to be taken into consideration.

Table 2: Breakdown data

Resource	**Downtime**	**Events/month**
Weld robots	2%	5
Manipulators	2%	3

Table 3: Shift patterns; Monday – Thursday

Start time	End time	Description
Shift 1		
0800hrs	0810hrs	Brief
0810hrs	0925hrs	Production
0925hrs	0935hrs	Break
0935hrs	1200hrs	Production
1200hrs	1230hrs	Lunch break
1230hrs	1425hrs	Production
1425hrs	1435hrs	Break
1435hrs	1630hrs	Production
Shift 2		
1630hrs	1640hrs	Brief
1640hrs	1825hrs	Production
1825hrs	1835hrs	Break
1835hrs	2025hrs	Production
2025hrs	2055hrs	Dinner break
2055hrs	2255hrs	Production
2255hrs	2305hrs	Break
2305hrs	0055hrs	Production
Shift 3		
0055hrs	0105hrs	Brief
0105hrs	0400hrs	Production
0400hrs	0420hrs	Dinner break
0420hrs	0800hrs	Production

Table 4: Shift patterns; Friday

Start time	End time	Description
Shift 1		
0800hrs	0810hrs	Brief
0810hrs	0925hrs	Production
0925hrs	0935hrs	Break
0935hrs	1230hrs	Production
Shift 2		
1230hrs	1240hrs	Brief
1240hrs	1430hrs	Production
1430hrs	1440hrs	Break
1440hrs	1655hrs	Production
Shift 3		
1655hrs	1705hrs	Brief
1705hrs	1900hrs	Production
1900hrs	1910hrs	Break
1910hrs	2115hrs	Production
2115hrs	2145hrs	Dinner break
2145hrs	0135hrs	Production

4 Simulation requirements

As ITC exports its trucks all over the world, there are no seasonal variations in terms of numbers of orders placed. Also, the sales forecast is updated on a 3-monthly basis, and therefore a simulation length of 3 months, with 5 independent simulation runs, is considered adequate. The warm-up period amounts to 1 week.

The management of ITC wants a simulation complete with animation to obtain the following output data:

- Utilization rate, idle time for:
  - Cranes (individually)
  - Assembly operators
  - Assembly jig
  - Weld robots
  - Weld manipulators (jig holding the parts during welding)
  - Welders
  - Liner fitting operator
  - Liner fitting jig
  - Output buffers
- Downtimes[1] for the weld robot
- Average time in system

You, as the simulation consultant, are tasked with the creation of the simulation model in Arena with the basic set of data indicated in section 3 on page 4; the complete set of experiments will be run by people within ITC.

The animation should be representative of the processes in assembly, but the small stationary crane in the liner fitting station do not need to be modelled.

Cranes 1 through 4 can be modelled as transporters or resources.

Due to the required output data, assembly jigs and weld manipulators will have to be modelled as resources.

Please document the model, as it will be passed on to ITC for experimentation.

[1] MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair) put together

Validation Simulation – Part B

Tom Pohl
Sheffield Hallam University

August 16, 2006

1 Introduction

ITC (International Truck Corporation) produces 2 types of dump trucks, large and small ones, and in each case, there is a choice between trucks with and without liners. — The liners offer better protection against wear and tear when certain abrasive materials are transported in ITC vehicles.

Management has decided to have a simulation model of their truck floor production line built, to evaluate the optimum number of operators, and to see whether there is a potential bottleneck somewhere.

2 System Description

2.1 Layout and Processes

The layout in figure 1 on the following page shows the truck floor production line. The individual parts are delivered from the left into the respective parts buffers via dedicated transporters.

Depending on current orders, crane 1 or crane 2 picks up the required parts and delivers them to either the large or the small assembly station, where they are manually prepared for machine welding.

Once the truck floor is set up for welding, the assembled parts are transferred to the corresponding weld station via crane 3, where the weld robot then processes them.

After welding, crane 3 then picks up the welded truck floor and delivers it either to one of 5 output buffers or to the liner fitting station.

Once the truck floor has had the liner plates affixed – if so requested, it is either moved out of the production line with crane 4, or transferred to one of the output buffers, from which crane 4 then transfers them onto a dedicated transporter to move them on to the final truck assembly.

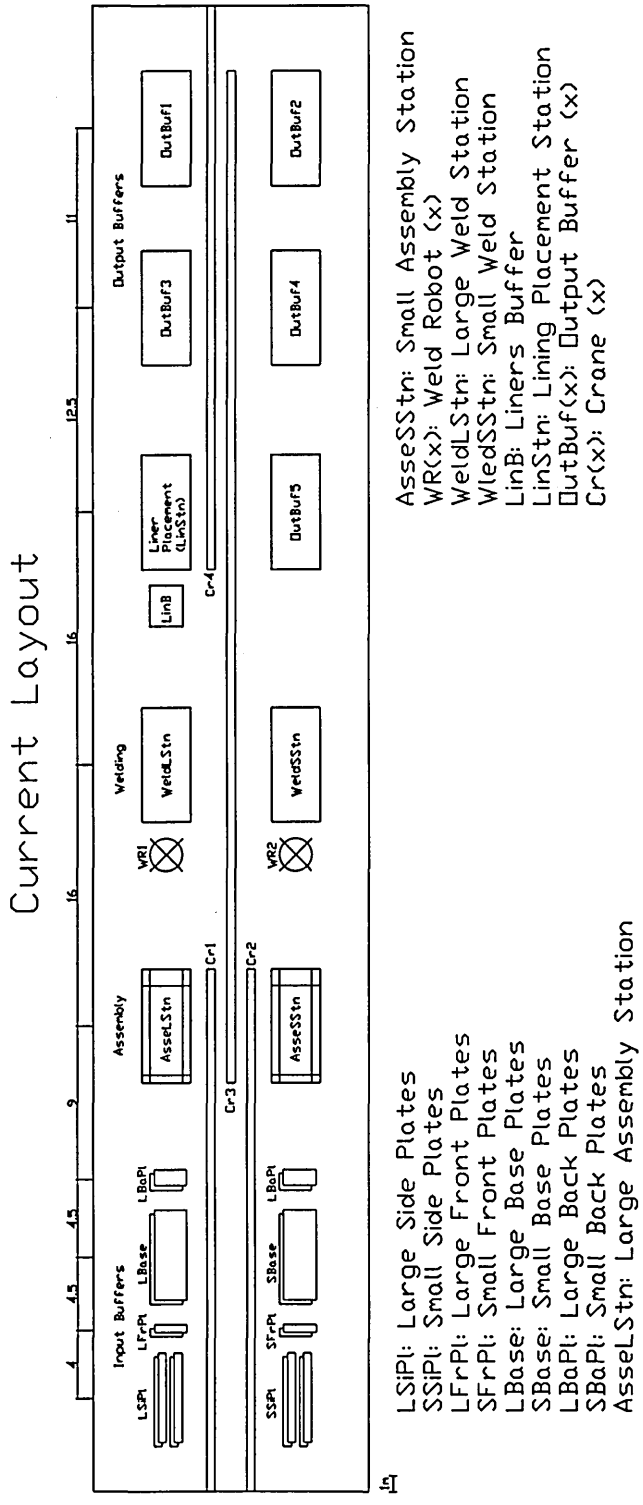

Figure 1: Current layout

2.2 Assumptions

2.2.1 Buffers

All buffers (base plates, front plates, back plates, side plates and liner plates) are regularly restocked from outside and do not run below a critical level.

As they are restocked from outside, by employees from a different part of the factory, this activity has no influence on the workload of the assembly line workers.

2.2.2 Cranes

In order to get the parts from one end of the production line to the other, the cranes have to overlap to some extent. However, they are all placed at different heights and do therefore not obstruct each other.

The cranes are operated by the assembly line workers who need them, e.g. by the assembly operators to put the parts together, by the welders to pick the assembled truck floor up for welding etc. I.e., all assembly workers are trained in crane operation.

There are two cranes available to assembly (cranes 1 and 2), three for the transfer between the assembly and welding (only one, crane 3, is shown in the diagram) and one crane to unload the liner fitting station and take care of the output buffers.

2.2.3 Assembly/Pre-weld

At the assembly stage, a worker spot-welds the parts together, which are held in place by crane 1 or crane 2 during spot welding.

Therefore, a worker can only either get a part for assembly, or spot-weld parts together.

2.2.4 Welding

While the majority of welding is done by the weld robot, some rework needs to be done manually, and a worker has to be present to monitor the welding process.

While there is one robot dedicated to large truck floors and one dedicated to small ones, the weld robots themselves are identical and only run slightly different programs. Changing the set-up from welding small truck floors to welding large ones, or vice-versa, takes 5min $\pm 10\%$.

2.2.5 Liner plate fitting

The placement of liner plates is done manually, helped by a small, stationary crane (not shown in figure 1 on the previous page).

2.2.6 Output buffers

There are currently 5 output buffers, and unloading those takes priority over unloading a weld or liner placement station, as the arrival of transporters to pick up finished truck

floors is determined by outside factors, and is sometimes a little sporadic. On average, a truck floor spends 5 hours±10% in the buffers.

3 Data

Orders for the assembly of truck floors are received roughly every 140 minutes (exponential distribution).

The current ratio of orders for small versus large trucks is 50% to 50%, with 30% of small trucks and 10% of large trucks requiring the addition of liner plates.

Table 1 shows the loading, unloading and process times. The loading/unloading times contain the cranes' travel time. The cranes' maximum speed under load is $0.5ms^{-1}$, empty $1.5ms^{-1}$.

Table 1: Load, unload and process times (in minutes)

	Small			Large		
Load time assembly	20	±	20%	20	±	20%
Assembly	98	±	5%	100	±	5%
Unload time (generic)	5	±	20%	5	±	20%
Load time (generic)	2	±	20%	2	±	20%
Weld time	46	±	2.5%	48	±	2.5%
Liner fitting time	89	±	2.5%	112	±	2.5%

The employees at ITC work in 3 shifts from Monday to Friday, although the Friday shifts are different from the rest of the week. Tables 3 on the next page and 4 on the following page show the specific shift patterns.

Per shift, there is one worker at each of the assembly stations, one at each weld station, one at the liner plate fitting station, and one who handles the unloading of the weld/liner fitting stations, loading and unloading of the output buffers. The operators have an efficiency of 80%.

Table 2 gives an overview of breakdown patterns for weld robots and manipulators. The other resources are assumed to be of so good a maintenance that breakdowns for them do not need to be taken into consideration.

Table 2: Breakdown data

Resource	**Downtime**	**Events/month**
Weld robots	2%	5
Manipulators	2%	3

Table 3: Shift patterns; Monday – Thursday

Start time	End time	Description
Shift 1		
0800hrs	0810hrs	Brief
0810hrs	0925hrs	Production
0925hrs	0935hrs	Break
0935hrs	1200hrs	Production
1200hrs	1230hrs	Lunch break
1230hrs	1425hrs	Production
1425hrs	1435hrs	Break
1435hrs	1630hrs	Production
Shift 2		
1630hrs	1640hrs	Brief
1640hrs	1825hrs	Production
1825hrs	1835hrs	Break
1835hrs	2025hrs	Production
2025hrs	2055hrs	Dinner break
2055hrs	2255hrs	Production
2255hrs	2305hrs	Break
2305hrs	0055hrs	Production
Shift 3		
0055hrs	0105hrs	Brief
0105hrs	0400hrs	Production
0400hrs	0420hrs	Dinner break
0420hrs	0800hrs	Production

Table 4: Shift patterns; Friday

Start time	End time	Description
Shift 1		
0800hrs	0810hrs	Brief
0810hrs	0925hrs	Production
0925hrs	0935hrs	Break
0935hrs	1230hrs	Production
Shift 2		
1230hrs	1240hrs	Brief
1240hrs	1430hrs	Production
1430hrs	1440hrs	Break
1440hrs	1655hrs	Production
Shift 3		
1655hrs	1705hrs	Brief
1705hrs	1900hrs	Production
1900hrs	1910hrs	Break
1910hrs	2115hrs	Production
2115hrs	2145hrs	Dinner break
2145hrs	0135hrs	Production

4 Simulation requirements

As ITC exports its trucks all over the world, there are no seasonal variations in terms of numbers of orders placed. Also, the sales forecast is updated on a 3-monthly basis, and therefore a simulation length of 3 months, with 5 independent simulation runs, is considered adequate. The warm-up period amounts to 1 week.

The management of ITC wants a simulation complete with animation to obtain the following output data:

- Utilization rate, idle time for:
  - Cranes (individually)
  - Assembly operators
  - Assembly jig
  - Weld robots
  - Weld manipulators (jig holding the parts during welding)
  - Welders
  - Liner fitting operator
  - Liner fitting jig
  - Output buffers
- Downtimes[1] for the weld robot
- Average time in system

You, as the simulation consultant, are tasked with the creation of the simulation model in Arena with the basic set of data indicated in section 3 on page 4; the complete set of experiments will be run by people within ITC.

The animation should be representative of the processes in assembly, but the small stationary crane in the liner fitting station do not need to be modelled.

Cranes 1 through 4 can be modelled as transporters or resources.

Due to the required output data, assembly jigs and weld manipulators will have to be modelled as resources.

As the completed simulation project will be passed on, please read through and follow the guidelines (see attachment) before and during the creation of the simulation model for ITC.

[1] MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair) put together

Validation Simulation – Part C

Tom Pohl
Sheffield Hallam University

September 17, 2006

1 Introduction

ITC (International Truck Corporation) produces 2 types of dump trucks, large and small ones, and in each case, there is a choice between trucks with and without liners. — The liners offer better protection against wear and tear when certain abrasive materials are transported in ITC vehicles.

As an increase in orders has been forecast, the management is considering the purchase of two additional manipulator jigs, and an update of their weld robots. The two new manipulator jigs would be identical to the two existing ones, and would replace the assembly jigs. This would further eliminate the transport of the assembled truck floors to the weld area, as the assembly would take place on the weld manipulator.

The update of the weld robots would substantially reduce weld time. Furthermore, the updated robots could handle the work on two jigs, alternating between the two.

Management are also considering the addition of a second jig to fit liner plates, and a dedicated crane for this work.

From these modifications, ITC are expecting an increased output to cope with the forecast increase in demand for ITC trucks.

2 System Description

2.1 Layout and Processes

The layout in figure 1 on the following page shows the current truck floor production line. The individual parts are delivered from the left into the respective parts buffers via dedicated transporters. The layout in figure 2 on page 3 shows the planned layout, with the additional manipulators.

2.1.1 Current situation

Depending on current orders, crane 1 or crane 2 picks up the required parts and delivers them to either the large or the small assembly station, where they are manually prepared for machine welding.

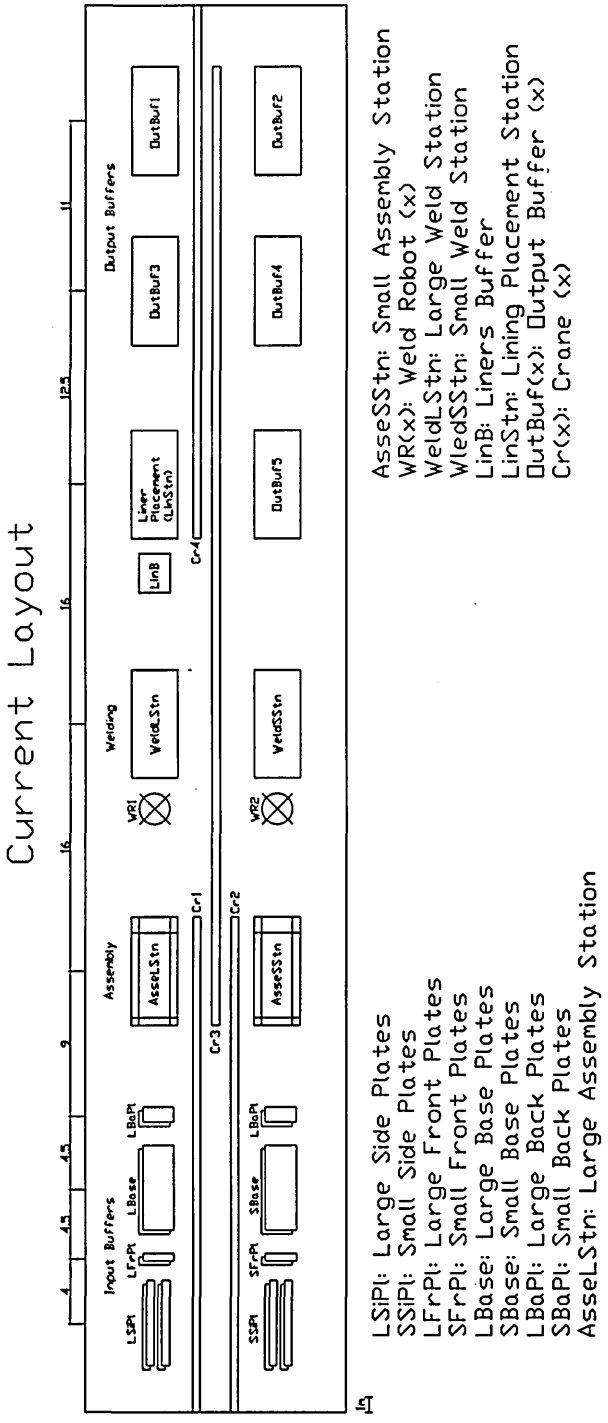

Figure 1: Current layout

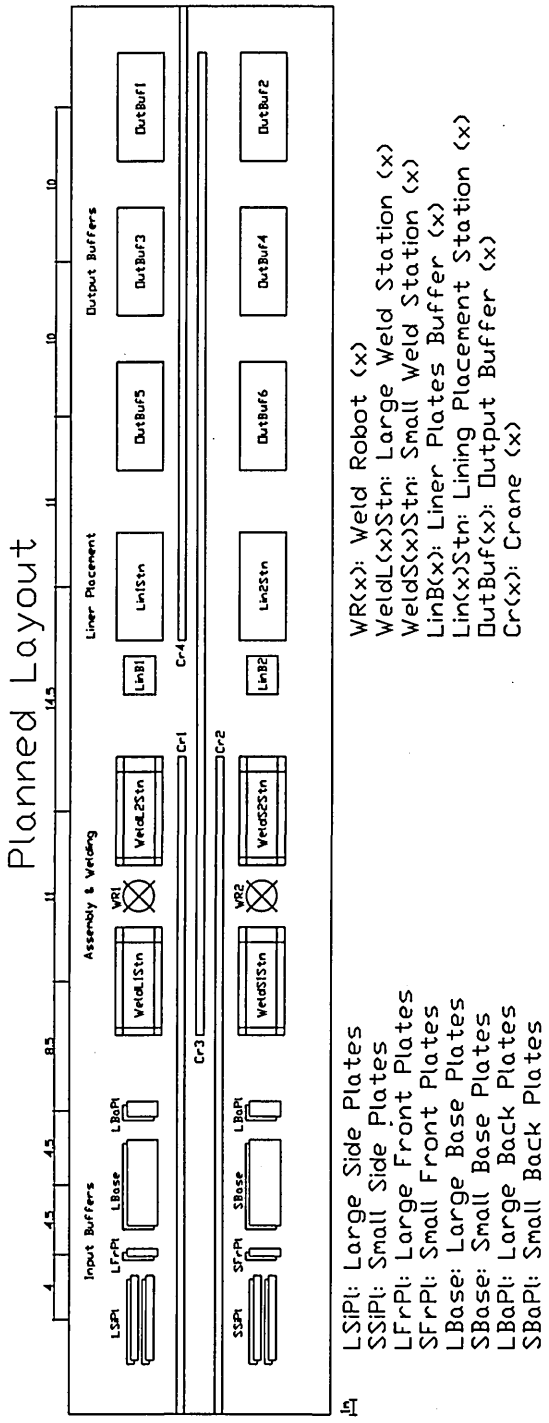

Figure 2: Planned layout

Once the truck floor is set up for welding, the assembled parts are transferred to the corresponding weld station via crane 3, where the weld robot then processes them.

After welding, crane 3 then picks up the welded truck floor and delivers it either to one of 5 output buffers or to the liner fitting station.

Once the truck floor has had the liner plates affixed – if so requested, it is either moved out of the production line with crane 4, or transferred to one of the output buffers, from which crane 4 then transfers them onto a dedicated transporter to move them on to the final truck assembly.

2.1.2 Future situation

When an order for a large truck comes in, crane 2 picks up the parts and delivers them to one of two assembly/weldstations dedicated to large trucks, where they are assembled *and* welded.

In case of an order for a small truck, it is crane 1 that picks up the parts and delivers them to one of two assembly/weldstations dedicated to small trucks, where they are assembled *and* welded.

If one of the two weld robots breaks down, the two assembly/weldstations handled by that robot are not used, and all orders, for large and small trucks, goes to the other two assembly/weldstations.

Once the truck floor is welded, it is either passed on to one of two *identical* liner fitting stations, or it goes into one of six *identical* output buffers, starting by placing the truck floors in the buffers closest to the exit, i.e. on the right hand side. All this is done by one of units of crane 3.

Crane 4 only serves to unload the liner fitting stations, from where it transfers the truck floors to the output buffers, and to load the truck floors from the output buffers onto a dedicated transporter system so they can be transferred on to the final truck assembly.

2.2 Assumptions

2.2.1 Buffers

All buffers (base plates, front plates, back plates, side plates and liner plates) are regularly restocked from outside and do not run below a critical level.

As they are restocked from outside, by employees from a different part of the factory, this activity has no influence on the workload of the assembly line workers.

2.2.2 Cranes

In order to get the parts from one end of the production line to the other, the cranes have to overlap to some extent. However, they are all placed at different heights and do therefore not obstruct each other.

The cranes are operated by the assembly line workers who need them, e.g. by the assembly operators to put the parts together, by the welders to pick the assembled truck floor up for welding etc. I.e., all assembly workers are trained in crane operation.

There are two cranes available to assembly/welding (cranes 1 and 2), three for the transfer between the assembly/welding and liner fitting/buffers (only one crane, crane 3, is shown in the diagram) and one crane to unload the liner fitting station and take care of the output buffers.

2.2.3 Assembly/Pre-weld

At the assembly stage, a worker spot-welds the parts together, which are held in place by crane 1 or crane 2 during spot welding.

Therefore, a worker can only either get a part for assembly, or spot-weld parts together.

2.2.4 Welding

While the majority of welding is done by the weld robot, some rework needs to be done manually, and a worker has to be present to monitor the welding process.

While there is one robot dedicated to large truck floors and one dedicated to small ones, the weld robots themselves are identical and only run slightly different programs. Changing the set-up from welding small truck floors to welding large ones, or vice-versa, takes 5min ±10%.

Future situation At the assembly/weld stage, the same worker who works on the assembly in the fitting jig then also controls the weld robot and does the post-weld work, as well as manipulating the crane to transport parts from the input buffers to the assembly/weld stations. The other assumptions relating to assembly and welding from the current situation remain valid.

2.2.5 Liner plate fitting

The placement of liner plates is done manually, helped by a small, stationary crane (not shown in figures 1 or 2).

For the planned layout, it is expected that both liner fitting stations can process small as well as large truck floors.

2.2.6 Output buffers

Unloading of output buffers takes priority over unloading a weld or liner placement station, as the arrival of transporters to pick up finished truck floors is determined by outside factors, and is sometimes a little sporadic. On average, a truck floor spends 5 hours±10% in the buffers.

3 Data

3.1 Current situation

Orders for the assembly of truck floors are received roughly every 140 minutes (exponential distribution).

The current ratio of orders for small versus large trucks is 50% to 50%, with 30% of small trucks and 10% of large trucks requiring the addition of liner plates.

Table 1 shows the loading, unloading and process times. The loading/unloading times contain the cranes' travel time. The cranes' maximum speed under load is $0.5ms^{-1}$, empty $1.5ms^{-1}$.

Table 1: Load, unload and process times (in minutes)

	Small			Large		
Load time assembly	20	±	20%	20	±	20%
Assembly	98	±	5%	100	±	5%
Unload time (generic)	5	±	20%	5	±	20%
Load time (generic)	2	±	20%	2	±	20%
Weld time	46	±	2.5%	48	±	2.5%
Liner fitting time	89	±	2.5%	112	±	2.5%

The employees at ITC work in 3 shifts from Monday to Friday, although the Friday shifts are different from the rest of the week. Table 2 on the next page shows the specific shift patterns.

Per shift, there is one worker at each of the assembly stations, one at each weld station, one at the liner plate fitting station, and one who handles the unloading of the weld/liner fitting stations, loading and unloading of the output buffers. The operators have an efficiency of 80%.

Table 3 on the following page gives an overview of breakdown patterns for weld robots and manipulators. The other resources are assumed to be of so good a maintenance that breakdowns for them do not need to be taken into consideration.

Table 2: Shift patterns

Monday – Thursday

Start time	End time	Description
Shift 1		
0800hrs	0810hrs	Brief
0810hrs	0925hrs	Production
0925hrs	0935hrs	Break
0935hrs	1200hrs	Production
1200hrs	1230hrs	Lunch break
1230hrs	1425hrs	Production
1425hrs	1435hrs	Break
1435hrs	1630hrs	Production
Shift 2		
1630hrs	1640hrs	Brief
1640hrs	1825hrs	Production
1825hrs	1835hrs	Break
1835hrs	2025hrs	Production
2025hrs	2055hrs	Dinner break
2055hrs	2255hrs	Production
2255hrs	2305hrs	Break
2305hrs	0055hrs	Production
Shift 3		
0055hrs	0105hrs	Brief
0105hrs	0400hrs	Production
0400hrs	0420hrs	Dinner break
0420hrs	0800hrs	Production

Friday

Start time	End time	Description
Shift 1		
0800hrs	0810hrs	Brief
0810hrs	0925hrs	Production
0925hrs	0935hrs	Break
0935hrs	1230hrs	Production
Shift 2		
1230hrs	1240hrs	Brief
1240hrs	1430hrs	Production
1430hrs	1440hrs	Break
1440hrs	1655hrs	Production
Shift 3		
1655hrs	1705hrs	Brief
1705hrs	1900hrs	Production
1900hrs	1910hrs	Break
1910hrs	2115hrs	Production
2115hrs	2145hrs	Dinner break
2145hrs	0135hrs	Production
~	~	~

Table 3: Breakdown data

Resource	**Downtime**	**Events/month**
Weld robots	2%	5
Manipulators	2%	3

3.2 Modified data

While the ratio of orders for small versus large trucks remains at 50% to 50%, the amount of truck floor requiring lining increases to 35% for small trucks and to 20% for large trucks.

Due to an improvement to the weld robots and the combination of assembly and welding in one location, the process times change somewhat; table 4 gives an overview of these times for the modified setup.

Table 4: Load, unload and process times (in minutes)

	Small			Large		
Load time assembly	20	±	20%	20	±	20%
Assembly/weld	138	±	4%	143	±	4%
Unload time (generic)	5	±	20%	5	±	20%
Load time (generic)	2	±	20%	2	±	20%
Liner fitting time	89	±	2.5%	112	±	2.5%

An update to the weld robots further results in a reduction of breakdown events from *5* down to *4* per month, while changing the set-up for the weld robots, which so far took 5 minutes, now only takes 4 minutes. (Cf § 2.2.4 on page 5).

The management at ITC has also decided to introduce a new shift pattern, as shown in table 5. Lunch/dinner breaks are only *45 minutes* per employee. Per sector, only one person is allowed to go on break, while the other takes care of both stations. E.g., if one assembly worker goes on his lunch break, the other has to supervise both weld robots. The schedule for lunch/dinner breaks can be found in table 6 on the following page.

Small breaks (of five minutes) are allowed once every full hour, again for one person per sector only.

Table 5: Shift patterns – new

Start time	End time	Description
Monday		
0600hrs	1405hrs	Shift 1
1355hrs	2205hrs	Shift 2
2155hrs	0605hrs	Shift 3
Tuesday – Thursday		
0555hrs	1405hrs	Shift 1
1355hrs	2205hrs	Shift 2
2155hrs	0605hrs	Shift 3
Friday		
0600hrs	1405hrs	Shift 1
1355hrs	2200hrs	Shift 2

The shifts are rotated on a weekly basis, so that every group only gets to do one week of night shifts out of 3. That is, for 3 groups of employees, called groups A, B and C, the work pattern shown in table 7 on the next page arises.

Table 6: Lunch/dinner breaks

Start time	End time	Shift
1000hrs	11300hrs	1
1800hrs	1930hrs	2
0200hrs	0330hrs	3

Table 7: Shift rotation

Group	Week 1	Week 2	Week 3
A	Shift 1	Shift 3	Shift 2
B	Shift 2	Shift 1	Shift 3
C	Shift 3	Shift 2	Shift 1

4 Simulation requirements

As ITC exports its trucks all over the world, there are no seasonal variations in terms of numbers of orders placed. Also, the sales forecast is updated on a 3-monthly basis, and therefore a simulation length of 3 months, with 5 independent simulation runs, is considered adequate. The warm-up period amounts to 1 week.

The management of ITC wants a simulation complete with animation to obtain the following output data:

- Utilization rate, idle time for:
  - Cranes (individually)
  - Assembly operators
  - Assembly jig
  - Weld robots
  - Weld manipulators (jig holding the parts during welding)
  - Welders
  - Liner fitting operator
  - Liner fitting jig
  - Output buffers
- Downtimes[1] for the weld robot
- Average time in system

You, as the simulation consultant, are tasked with the creation of the simulation model in Arena with the basic set of data indicated in section 3 on page 6; the complete set of experiments will be run by people within ITC.

The animation should be representative of the processes in assembly, but the small stationary crane in the liner fitting station do not need to be modelled.

Cranes 1 through 4 can be modelled as transporters or resources.

[1] MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair) put together

Due to the required output data, assembly jigs and weld manipulators will have to be modelled as resources.

Please document the model, as it will be passed on to ITC for experimentation.

5 Evaluation Form for Part A – without guidelines

		Very easy	Somewhat easy	Somewhat difficult	Very difficult
ADD	Product (entity)				
	Process/resource/ Work center				
	Handling device				
	Sequence				
	Shift				
	Breakdown data				
	Attributes				
	Variables				
MODIFY	Product (entity)				
	Process/resource/ Work center				
	Handling device				
	Sequence				
	Shift				
	Breakdown data				
	Attributes				
	Variables				
DELETE	Product (entity)				
	Process/resource/ Work center				
	Handling device				
	Sequence				
	Shift				
	Breakdown data				
	Attributes				
	Variables				

6 Evaluation Form for Part B – with guidelines

		Very easy	Somewhat easy	Somewhat difficult	Very difficult
ADD	Product (entity)				
	Process/resource/ Work center				
	Handling device				
	Sequence				
	Shift				
	Breakdown data				
	Attributes				
	Variables				
MODIFY	Product (entity)				
	Process/resource/ Work center				
	Handling device				
	Sequence				
	Shift				
	Breakdown data				
	Attributes				
	Variables				
DELETE	Product (entity)				
	Process/resource/ Work center				
	Handling device				
	Sequence				
	Shift				
	Breakdown data				
	Attributes				
	Variables				

1 Part A

The model shown in figure 1 on page ii, built without use of the *modelbuilding guidelines*, was returned to the author with these comments:

"The things which I've left out are:

- The robots failure time. I've added a failure time for them. But, they both happen at the same time. So, that would mean that there is no point in me trying to deviate the job to another robot when the first one is not available. You have specified in the question how many times the failures occur, but not how often the failures occur(like a count).
- The set up time for the robot. Again, the situation is the same as above. I've done a set up time for the first ever entity. But,for a changeover scenario , it will only need to occur when one of them is failed."

[Modeller A]

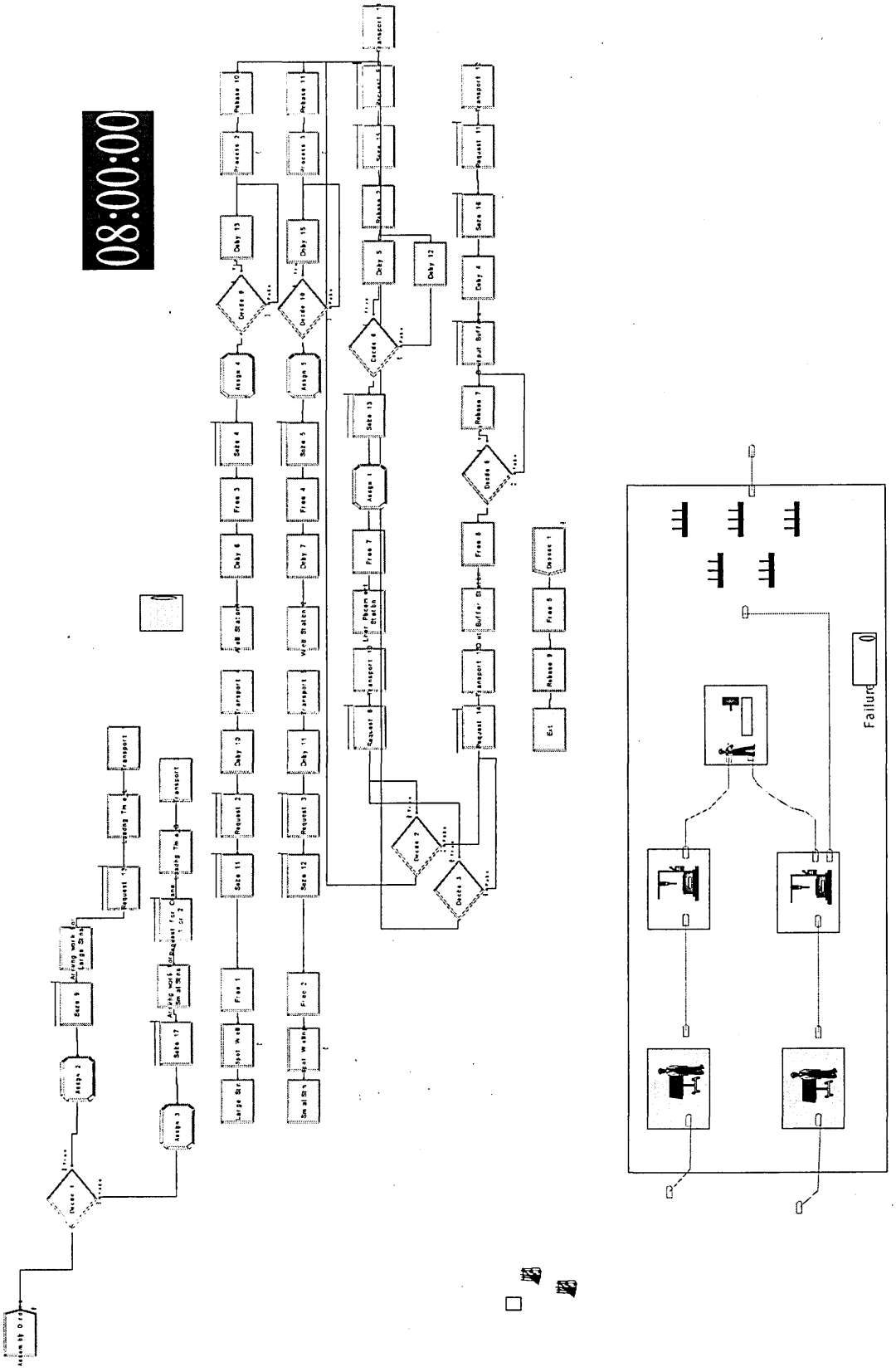

Figure 1: Snapshot of model, Part A

2 Part B

The model shown in figure 2 on page iv, built using the *modelbuilding guidelines*, was returned with the following comments:

> "The model seems to work although there is a problem with some resources/flow. Operator 6 and Crane 4, which are resources I used for unloading the Liner plate station and loading/unloading the output buffers, completely seize up during the run. I had worked out that if I make their capacities infinite it works, also if they don't have to unload the linear station and just deal with the output buffers its fine.
>
> The info given was pretty good although fairly difficult to follow the way you had laid out the data, I found myself sifting through pages, and making lots of changes as I was building the model, where when I read another bit of the paperwork I realised I had made wrong assumptions or misunderstood things etc.
>
> One definite assumption I had to make was which resources unloaded the weld stations, this is not stated.
>
> I haven't got around to the animation part or setting up statistics for the simulation requirements."
> *[Modeller B]*

Fragment of model source code, model logic (*.mod-file):

```
;
;
;     Model statements for module:  Station 1
;
130$            STATION,        AsseLStn;
168$            DELAY:          0.0,,VA:NEXT(84$);
;
;
;     Model statements for module:  Seize 10
;
84$             SEIZE,          2,Other:
                                LargeAssemblyJig,1:NEXT(170$);
170$            DELAY:          0.0,,VA:NEXT(5$);
;
;
;     Model statements for module:  Delay 6
;
5$              DELAY:          AssemblyLoadTime,,Other:NEXT(6$);
```

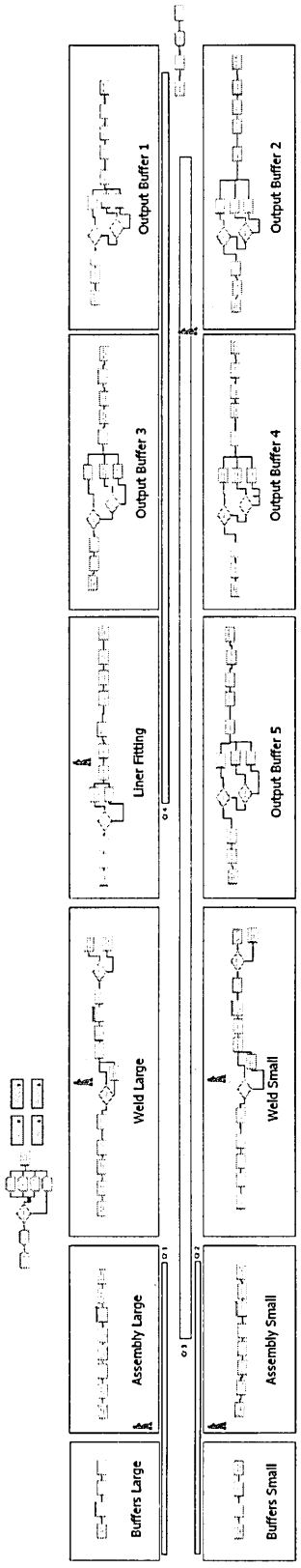

Figure 2: Snapshot of model, Part B

Fragment of model source code, input data (*.exp-file):

```
ATTRIBUTES:    TimeIn System:
               BufferTime:
               Route From Large Weld Stn_att:
               UnloadTime:
               Route From Small Weld Stn_att:
               AssemblyLoadTime:
               TimeOfArrival:
               weldrobot:
               LinearTime:
               linear:
               Route From Linear Stn_att:
               AssemblyTime:
               LoadTime:
               ChangeoverTime:
               WeldTime;

SCHEDULES:     Shift,TYPE(Capacity),FORMAT(Duration),
FACTOR(1.0),UNITS(minutes),DATA(0,490),DATA(1,75),
DATA(0,10),DATA(1,145),
```

Fragment of model source code, definition of resources and variables (*.opw-file):

```
[RESOURCES]
WeldRobot 1
WeldRobot 2
Operator 1
Operator 2
Operator 3
Operator 4
Operator 5
Operator 6
LargeWeldManipulator
Crane 1
Crane 2
Crane 3
Crane 4
SmallWeldManipulator
LinearJig
OutputBuffer 1
OutputBuffer 2
OutputBuffer 3
OutputBuffer 4
OutputBuffer 5
LargeAssemblyJig
SmallAssemblyJig
[VARIABLES]
Is WeldRobot 1 selected?.NumberOut False
Linear b1?.NumberOut False
Large Truck b1?.NumberOut True
```

3 Part C

The '*modifier*' ranked the – identical – changes to both models as shown in table 1, with 1 representing a *very difficult* modification and 4 a modification *very easy* to implement.

Table 1: Rating of ease of modification

	Part A	Part B
Add product (entity)	1	4
Add process/resource	2	3
Add handling device	3	2
Add sequence	2	3
Add shifts	3	3
Add attributes	1	4
Add variables	1	3
Modify product (entity)	2	4
Modify process/resource	1	4
Modify handling device	2	3
Modify shifts	3	3
Modify breakdown data	~	3
Modify attributes	1	4
Modify variables	1	4
Delete product (entity)	1	4
Delete process/resource	1	4
Delete handling device	1	2
Delete shifts	3	3
Delete breakdown data	~	4
Delete attributes	1	4
Delete variables	1	4

The modifier has also made the following comments regarding the ease of modifications:

> "The way to design and build these simulations are totally different i.e. the one without guideline (*Part A*) and another one with guideline (*Part B*). There are several points which can compare between these two

models:

- Processes Name

  Part B use meaning process name, therefore this is very easy to Add/Modify/Delete processes directly from the processes overview window without going through the whole model logic.

  Part A use default process name, therefore this is not possible to Add/Modify/Delete any of the process without going through the whole model logic.

- Entities Name

  Part B again use meaningful name to represent different entities. In addition, with the use of Attitudes and Variables for each entity from the first 'Input Data' view. This makes me very easy to add/modify/delete any of them; of course this is very easy to understand by the attitudes/variables name it used.

  However, *Part A* doesn't use any attitudes/variables attached with the entity, also the entity only represent by 'Type1' and 'Type2' which is difficult to understand either small plate or large plate they are represented.

- Model layout

  Part B utilise the name views function from ARENA to separate different views of the system, also with drawing a square around each process which make the model looks very clear and easy to follow. I don't really need to spend time on understand the model. However still in *Part B*, I can't see the use of sub-model in order to reuse same source of process. For example when I tried to add one more buffer to the system, I really need to duplicate the same code from others buffer and just modify the buffer name, delay time and attitude. However, if sub-model is used, I can just use SET function to simply add one more buffer with specific attitude without making the model looks too complicate and big.

  Part A is very different. When I first look at it, I really don't understand what is doing in this simulation model. I really need to spend double time to go through the whole model in order to know which process lines are representing. Because there is no label or name to specify what process is related. This made me very difficult to follow and do modification. However, as this model use SET to represent those buffers, I found it easier to add additional buffer without duplicate same coding.

Overall, I found that the *Part A* model is quite hard to remodel, comparatively *Part B* model is easier to understand and remodel."
[Modifier]

Figures 3 and 4 show snapshots of the finished, modified models.

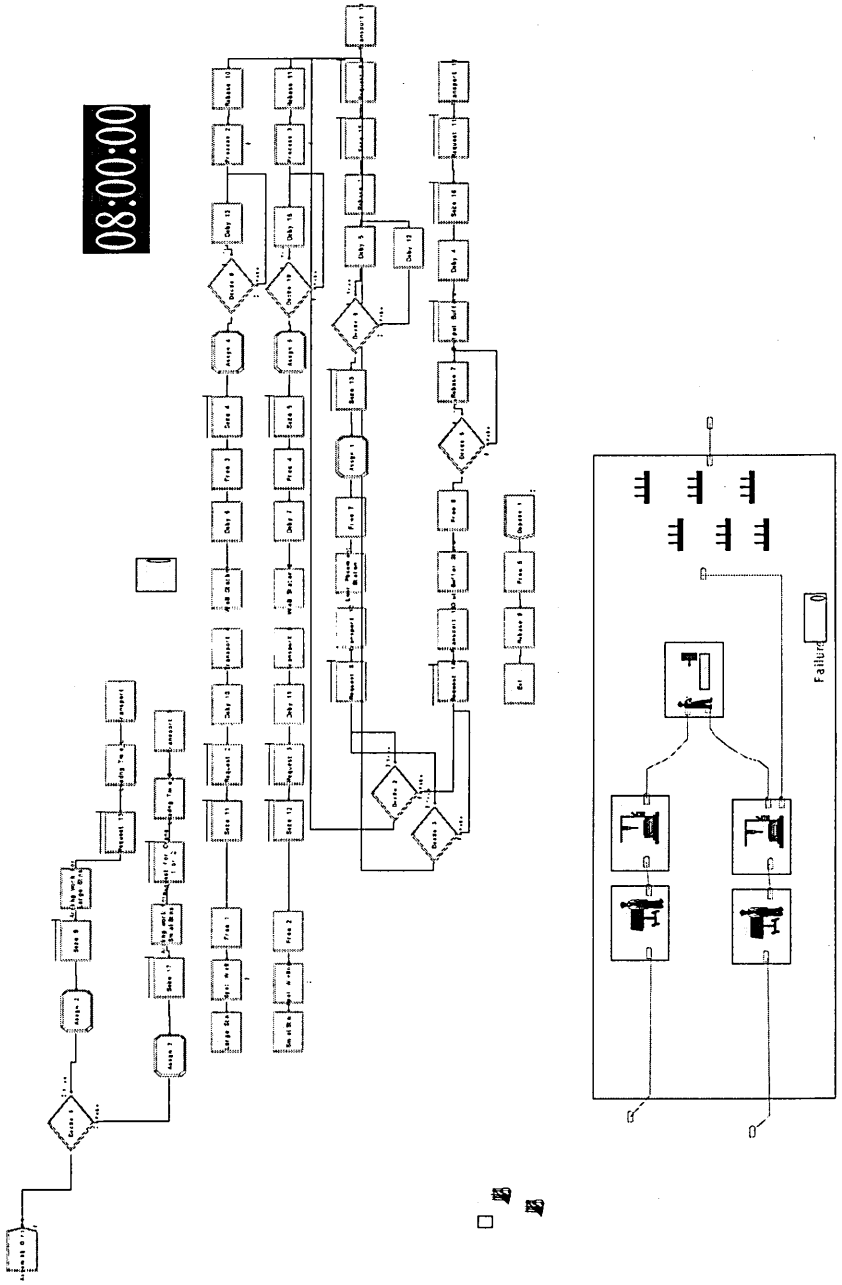

Figure 3: Snapshot of model, Part A – modified

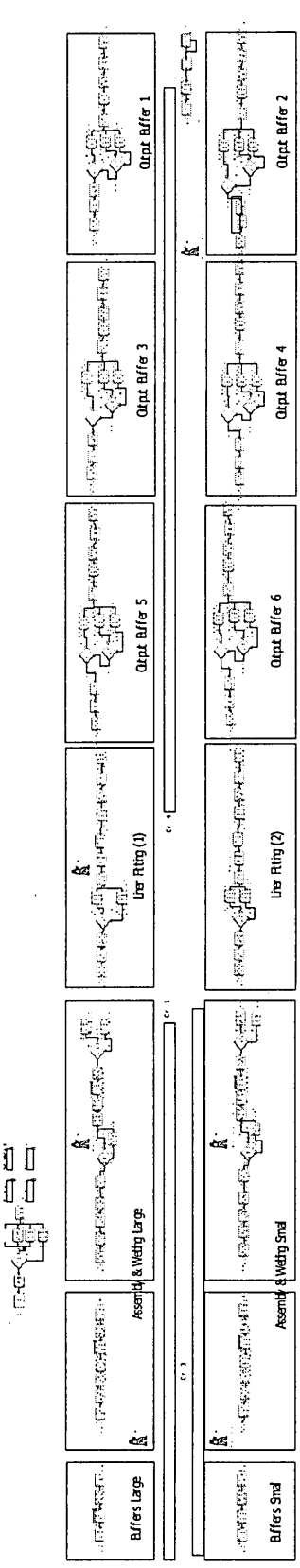

Figure 4: Snapshot of model, Part B – modified

Appendix G

Modified Model Building Guidelines

Modified Model Building Guidelines

T. Pöhl

1 Steps in a simulation study

Based on Law & Kelton (2000).

1. Formulate the problem and plan the study
   (a) Problem of interest is stated by manager
   (b) One or more kickoff meetings for the study are conducted, with the project manager, the simulation analysts, and subject-matter experts (SMEs) in attendance. The following issues are discussed:
      - Overall objectives of the study
      - Specific questions to be answered by the study
      - Performance measures that will be used to evaluate the efficacy of different system configurations
      - Scope of the model
      - System configurations to be modeled
      - Software to be used
      - Anticipate and plan for future modifications of model – through inclusion of higher management and SMEs
      - Time frame for the study and the required resources
2. Collect data and define the model
   (a) Collect information on the system layout and operating procedures
      - No single person or document is sufficient
      - some people may have inaccurate information – make sure that true SMEs are identified
      - Operating procedures may not be formalized
   (b) Collect data (if possible) to specify model parameters and input probability distributions
   (c) Delineate the above information and data in an "assumptions document," which is the *conceptual model.*
   (d) Collect data (if possible) on the performance of the existing system (for validation purposes in Step 6)
   (e) The level of model detail should depend on the following:
      - Project objectives
      - Performance measures
      - Data availability
      - Credibility concerns
      - Computer constraints

- Opinions of SMEs
- Time and money constraints

(f) There need not be a one-to-one correspondence between each element of the model and the corresponding element of the system

(g) Interact with the manager (and other key project personnel on a regular basis)

(h) Identify possibilities for future modification/reuse of the simulation model

3. Is the conceptual model valid?

   (a) Perform a structured walk-through of the conceptual model using the assumptions document before an audience of managers, analysts, and SMEs
   - Helps ensure that the model's assumptions are correct and complete
   - Promotes ownership of the model
   - Takes place *before* programming begins to avoid significant reprogramming later

4. Construct a computer program and verify

   (a) Program the model in a programming language (e.g., C or FORTRAN) or in simulation software (e.g., Arena, AutoMod, Extend, ProModel, WITNESS). Benefits of using a programming language are that one is often known, they have a low *purchase* cost, and they may result in a smaller model execution time. The use of simulation software, on the other hand, reduces programming time and results in a lower *project* cost.

   (b) Follow adaptability guidelines to provide for future reuse and modification of model

   (c) Verify (debug) the simulation computer program.

5. Does the programmed model provide for potential future modifications?

   (a) Do the adaptability options provide sufficient scope for later modification/reuse of the model?

   (b) Does the programmed model, in terms of potential adaptation, tie in to the company's – or department's – mid- to long-term strategy/plans?

6. Make pilot runs

   (a) Make pilot runs for validation purposes in Step 6

7. Is the programmed model valid?

   (a) If there is an existing system, then compare model and system (from Step 2) performance measures for the existing system

   (b) Regardless of whether there is an existing system, the simulation analysts and SMEs should review the model results for correctness

   (c) Use sensitivity analyses to determine what model factors have a significant impact on performance measures and, thus, have to be modeled carefully

8. Design experiments

   (a) Specify the following for each system configuration of interest:
   - Length of each run
   - Length of the warm up period, if one is appropriate

- Number of independent simulation runs using different random numbers – facilitates construction of confidence intervals

9. Make production runs
   (a) Production runs are made for use in Step 9
10. Analyze output data
   (a) Two major objectives in analyzing output data are:
      - Determining the absolute performance of certain system configurations
      - Comparing alternative system configurations in a relative sense
11. Document, present and use results
   (a) Document assumptions (see Step 2), computer program, and study's results for use in the current and future projects.
      - Document all assumptions relating to future modification/reuse of the model. – The assumptions made for the initial simulation might not hold up for the modified one, and should therefore be clearly stated in order to prevent the modified model from being inaccurate.
      - Create detailed documentation of computer program – e.g. of individual submodels or individual modules – to ease potential problems due to a change in modeler. If otherwise structures are unclear, a printout of the modules showing the logic connectors, or a flowchart, should be included.
   (b) Present study's results
      - Use animation to communicate model to managers and other people who are not familiar with all of the model details.
      - Discuss model building and validation process to promote credibility
   (c) Results are used in decision making process if they are *both* valid and credible
   (d) Document current simulation modeler/users and establish responsibilities for future projects
      - Which department holds documents and model
      - Prepare for current knowledge owner's absence, i.e. establish order of command/order of reference
   (e) Specifically document anticipated modifications and their scope to avoid unnecessary confusion when changes are to be implemented in the model

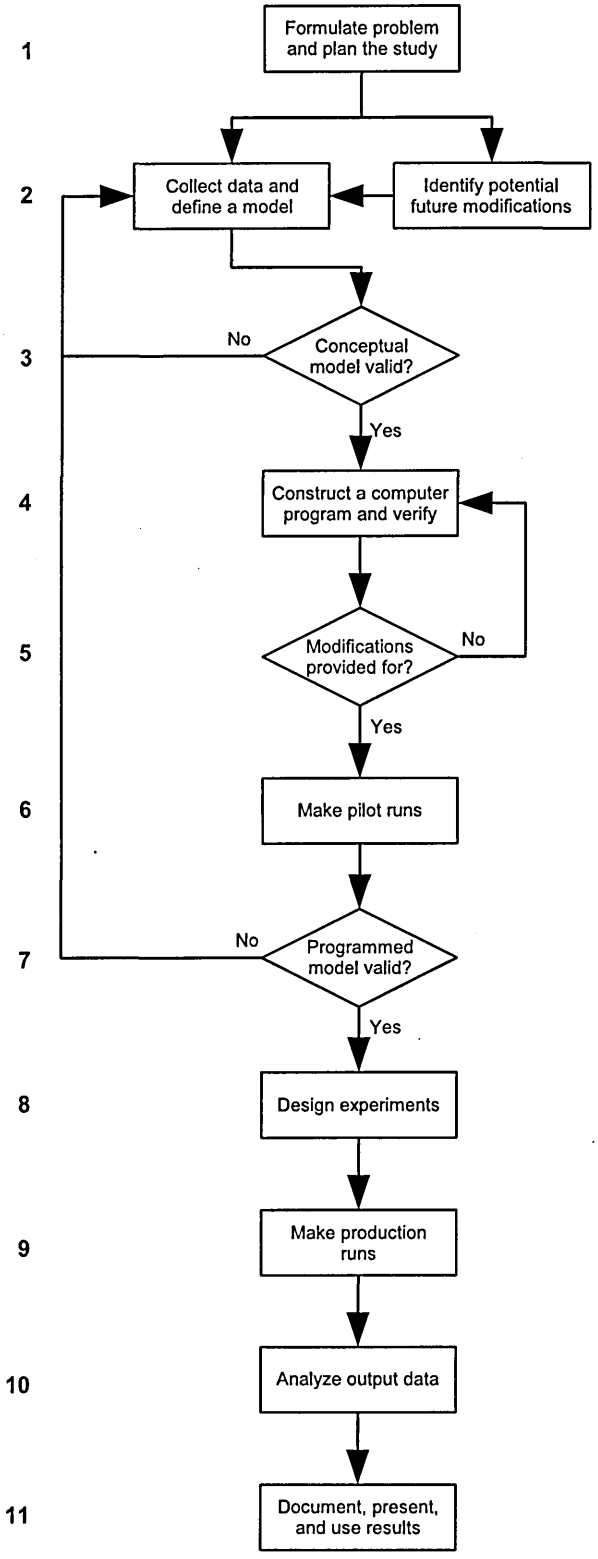

Figure 1: Steps in a simulation study, based on Law & Kelton, 2000, p. 84

2 The guidelines in detail

In this section, the elements added to the methodology are looked at in detail.

2.1 Formulate problem and plan the study

Traditionally, only the management directly affected by the simulation is involved in the whole process. Law & Kelton (2000), Balci (1990), Banks and Carson (1984) and Kelton et al. (2002) all point out the importance of involving the project manager and the management of the area affected by the simulation throughout the course of the study. However, as medium- to long-term strategy has a substantial influence on the whole company, and mainly senior management being involved in such medium- to long-term decisions, the author believes it is important for such higher management to be directly involved in, at least the initial stages of, the simulation study.

Another potential benefit of including higher management is that thereby they, too, assume some ownership of the simulation study, thus becoming more interested. I.e., this is a potential for improving the spread of simulation.

2.2 Identify potential future modifications

One of the main issues affecting the potential for software and -component as well as model reuse is the lack of explicitness of assumptions and constraints. In the case of potential future model modification, not only general assumptions should be made, but also assumptions regarding the type of changes to be anticipated in the future. If an organization's long-term strategy points to, e.g., diversification, then the potential influence of that strategy on the area under current study should be looked at and relating modifications planned for.

By concurrently defining the model *and* identifying potentials for modifications of the model that is being built, the requirement to make assumptions and constraints explicit is brought closer to realisation. As Spiegel et al. (2005) point out, capturing *all* assumptions and constraints is very unlikely to happen.

Also, as Kulick and Sawyer (1999) state, *"[t]he primary objectives of any simulation effort often determine the complexity of the tools needed"* (p. 1238). It is therefore of utmost importance to be aware, and plan for, future modifications, as this can – and will – impact on the complexity of the model.

2.3 Construct a computer program and verify

In order to incorporate the software-based provisions for adaptability, it is important to establish and follow specific adaptability guidelines. *Step 4* in figure 1 on the preceding page [*item 4.(b)* in the list] takes this into account.

Section 3 on page 8 treats those necessary elements to provide for improved adaptability in detail.

2.4 Provisions for modifications

Step 5 in figure 1 on the preceding page [*item 5* in the list] is a separate step, as verifying whether the coded model provides for future modifications beyond the scope of the planned simulation study is an issue independent from creating a working and accurate computer model. That is, making sure the programmed model allows for future modification is a question of validity, rather than of verifying/debugging the code.

The verification of the provision for adaptability consists of two steps, which should – again – be done concurrently:

1. Comparison of the programmed model against general adaptability guidelines

2. Comparison of the adaptability coded into the model against the medium- to long-term strategies of the organization under study. — This differs from *item 1.(a)* in so far, as *1.(a)* is only concerned with the short-term strategy, i.e. the immediately to be achieved goal of conducting a simulation experiment.

Although listed as a separate logical step, this verification would be done concurrently with the coding of the model.

2.5 Documentation, presentation and use of results

Document all assumptions relating to future modification/reuse of the model: The documentation as suggested by Law & Kelton only contains assumptions relating to the current study. However, in order to facilitate reuse of the model at a later stage, it is important that all further assumptions, which were made to incorporate adaptability (if there was a need for any such additional assumptions) are also documented.

Create detailed documentation of computer program – e.g. of individual sub-models or individual modules – to ease potential problems due to a change in modeler: Although documentation of the computer program forms an integral part of Law & Kelton's approach, the adaptability-specific parts of the model should be additionally indexed, or included in a separate documentation, so that, if a new study based on a modification of an existing model is to be conducted, the modeller(s) can quickly evaluate the fit of the model to the new circumstances. For this reason, the documentation should include a flowchart-like representation, if a printout of the model or submodels does not offer sufficient information in terms of structuring and the flow of elements or sequence of processes.

Although most literature points out the importance of documenting the simulation model, none of these authors go into detail about how to document the computer program.

Gass (1978), among others, states that "*documentation should commence at the very beginning of a project*" (p. 281), and that "*the 'lack' of documentation is one of the main reasons cited for model failures*" (p. 281) . Gass also suggests a complete documentation methodology, but is more specific about the documentation of the program itself. He suggests to document, among others:

- input and output data, and test case that have been run
- flow charts of the program
- operating instructions for the computer operator
- explanation of options available in the model
- a printout of the actual program code

In addition, to these documentation principles put forth by Gass, documentational aids provided by the program itself should be used. Simul8™ for example allows the addition of comments to most, if not all, elements, and Arena™ offers this option on, e.g. submodels.

Document current simulation modeler/users and establish responsibilities for future projects: While simulation projects are generally conducted within a team, all the team members will have access to the model and documentation. However, this step in the documentation is relevant when the team is very small, and/or to provide for the possibility that another, unrelated team, may have a use for the simulation project.

Specifically document anticipated modifications and their scope to avoid unnecessary confusion when changes are to be implemented in the model: This differs from *item 11.(a)* in so far as this is not concerned with assumptions, such as simplifications of certain input data. The anticipated modifications, in this case, refer to the changes or types of changes identified as the most likely ones in step *1.(b) – Anticipate and plan for future modifications of model, through inclusion of higher management and SMEs.*

3 Adaptability guidelines

The adaptability guidelines explained in this section are based on the results from the questionnaire survey and the benchmarking and influenced by the interviews conducted at an earlier stage.

Adaptability of simulation models for implementation of more substantial changes, several months or years after its last use, can be greatly aided by using the following upon building the initial model:

1. Sequencing
2. Model layout
3. In-built annotation possibilities
4. Type of data (internal vs external)

3.1 Sequencing

The use of sequencing may, at first, increase the complexity of the model somewhat (i.e. increase the number of steps needed to build the model), but has shown to simplify the adaptability of the model later on. This is because, contrary to hard-coding of all information, sequencing allows the user to specify the routes entities take in a more centralized way, e.g. by means of an internal database.

When hard-coding is used, for example after each resource where there are different directions an entity can take, to determine where that entity should go, many decision blocks have to be added. Those decision blocks would state that *"if the entity is of type* A*, then it should go to the left, otherwise, to the right"*, and one such block would be needed at every decision point.
When sequencing is used, however, these blocks need not be used, as per item there will be a list of resources, or work stationgs, that item has to go through. This means that some form of database, within the simulation package itself, is populated with data stating, for example, that *"entity of type* A *has to go through work stations* 1, 2, 3, 5 *and* 7*, while entity of type* B *has to go through work stations* 1, 3, 4, 6 *and* 7*"*. As a result, the person in charge of modifying the model only needs to look in one place to find out where elements are going, rather than having to look through the whole code.

However, as sequencing can cloud the flow of items or processes, a flowchart should be included as part of the documentation.

3.2 Model layout

Contrary to sequencing, which does decrease the number of programming steps at the time of implementing a change, the use the model layout has no such direct benefits. However, along with annotation, it helps the user understand and navigate the existing model, thereby reducing the time needed to understand and consequently modify the model. This, in turn, cuts down on the overall model-building and -modifying time and reduces the perceived need to build a new model from scratch.

It is best to avoid complex, tiered structures, such as submodels or templates. While they are helpful in reducing the number of elements visible at any one time, and therefore aid the basic, high-level understanding, they make it more difficult for someone other than the original simulation builder to modify the model. Graphical aids, e.g. grouping objects and using graphical tools, such as drawing boundaries around groups, also help in gaining a primary, high-level understanding, while at the same time granting uncomplicated low-level access, and should therefore be given priority over submodels (and/or custom-built templates in packages such as Arena).

Tiered structures result in the model consisting of different layers of varying detail. The top-level would only show the connection from submodel to submodel, and the user has to open the submodel to see how elements are routed within it, or what resources and handling devices there are. By only graphically grouping everything, for example by drawing a square around the modules that make up one work center, this helps the user to still differentiate between the different logic groupings, while still enabling that user to see all the connections which would otherwise be hidden.

Where available, *named views* (e.g. in Arena) should be used, as these also help in navigating the model. With named views, one specific area of the modelling surface, e.g. what would amount to one subgroup, can be assigned a view, to which the user can switch from, e.g. the overall model view.

3.3 Annotation

While submodels help organize the model in a logical way, thereby making it easier for the user/model builder to navigate through the simulation model, the annotation facilities built-in to most modern COTS simulation packages can be of great help in clarifying ambiguous terms (e.g. the use of specific variables can be explained in more detail or confusion prevented).

Where there is an option to output actual code (such as in Arena, with the *"write SIMAN"* command), this should also be added to complete the documentation/annotation.

Another highly important element to help with annotation – and understanding of the model itself – is the choice of names. All resources, attributes and variables should be given logic names that are easy to understand; e.g. attribute names such as *att1*, *att2* etc. should be avoided in favour of names such as *itmlgth* for the attribute *'item length'*. Also, attributes, variables and resources, as well as other elements of significance, should be stated as part of the documentation, with a short description, e.g. the name and what the element represents.

As mentioned in section 3.1 on the previous page, a flowchart of the processes should be included, especially if sequencing is used, as this can make understanding the structure more difficult.

3.4 Data type

Depending on the (estimated) amount of input data, the simulationist has to decide whether to include that data in the simulation, or put it into an external database, as a later change from internal to external tends to be quite cumbersome. This approach works well for simulation models with a large amount of input (or output) data, especially if that data already happens to be present in the form of a database or spreadsheet.

However, the use of externally stored data, e.g. in the form of a database, requires some VBA (or other programming language) knowledge and a good understanding of the simulation package.

The matrix in figure 2 on the following page is intended to aid in deciding whether or not to use the data – including estimates for future data requirements – internally or externally. In this case, *"small amount of data"* refers to the amount of work needed to include the data internally being substantially smaller than that needed to store the data externally and programming the interface. This *'amount of work'* could for example be measured in man-hours. As what constitutes 'large' or 'small' amounts of data is very subjective, no clear cut-off point for its placement within or outside the simulation model can be given at this point.

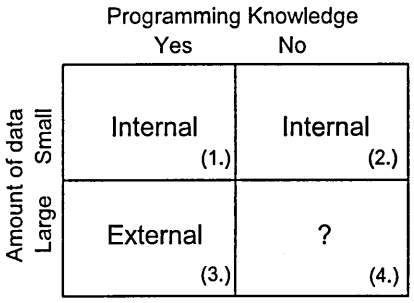

Figure 2: Data placement matrix

The matrix (fig. 2) in detail:

1. Small amount of data and programming knowledge: No need for storing data externally
2. Small amount of data, but no programming knowledge: No need for storing data externally
3. Large amount of data and programming knowledge: Use of external data storage strongly suggested, even though this means that any future model handler, who has to modify it, will need to have those specific programming skills
4. Large amount of data, but no programming knowledge: Use of external data storage still suggested, but only if model handler is willing and able (abilities as well as time allowance) to learn the necessary skills

However, it also has to be noted that, in the case of externally stored data, the requirements for documentation also increase; specifically, the documentation will have to include:

- Details about the external storage method (name and description of database/spreadsheet, caveats)
- Details about the interface (to enable modeler to understand and retrace the steps made during the first build phase and modify the code if and where necessary)

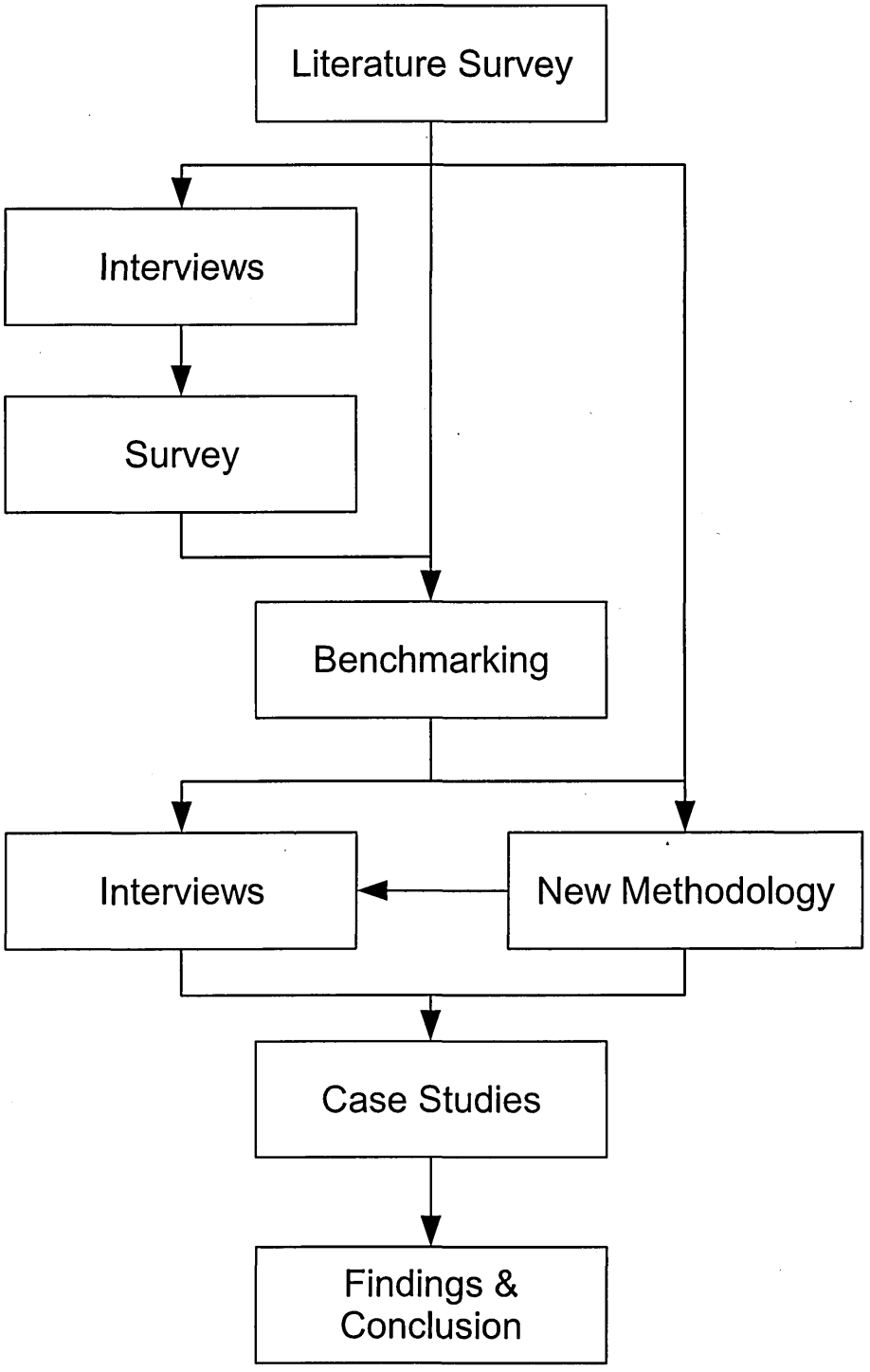

Figure 1: Key stages – flowchart

Chapter 4 –
Interviews and Questionnaire Survey

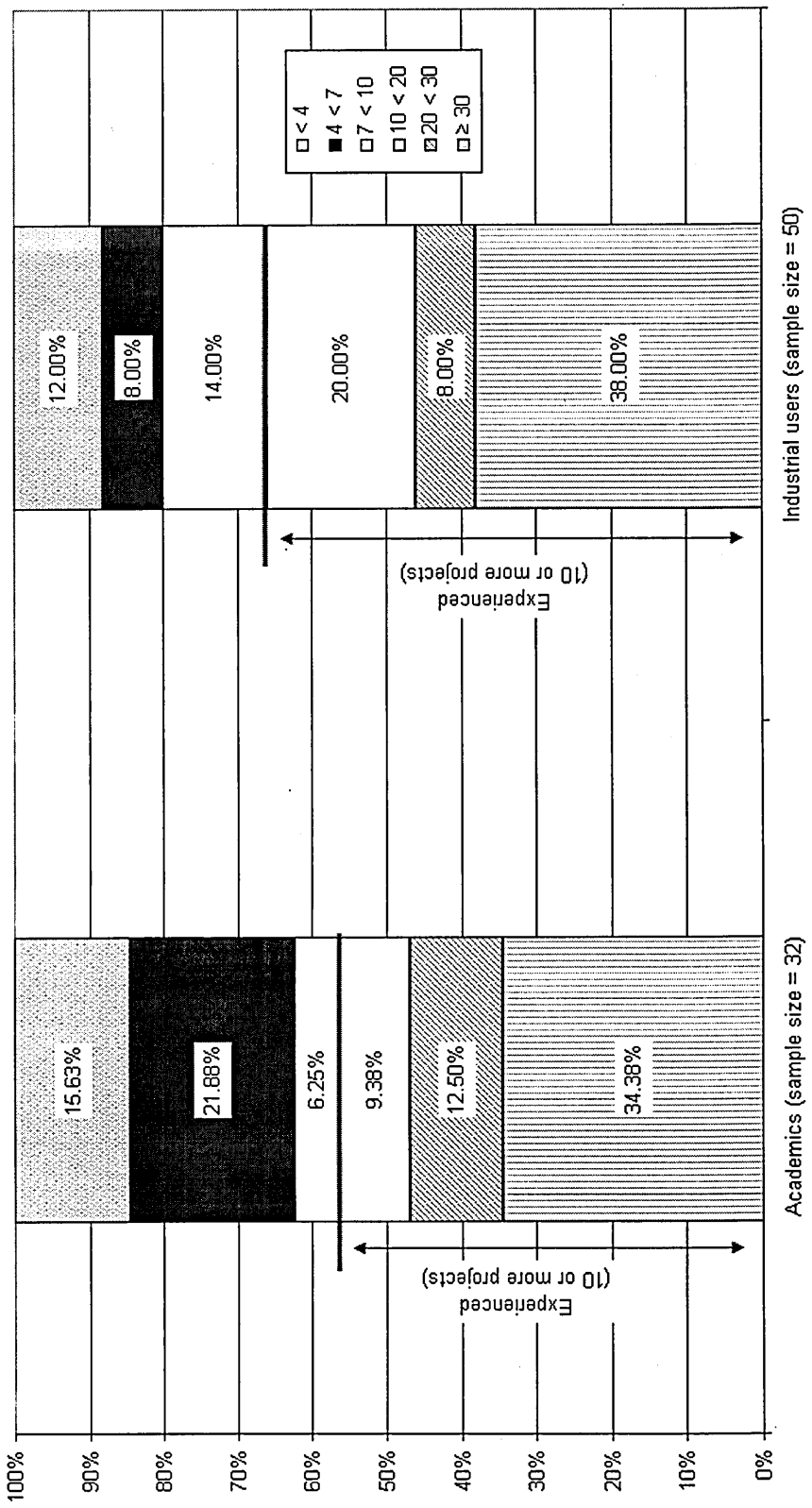

Figure 2: Overall model building experience

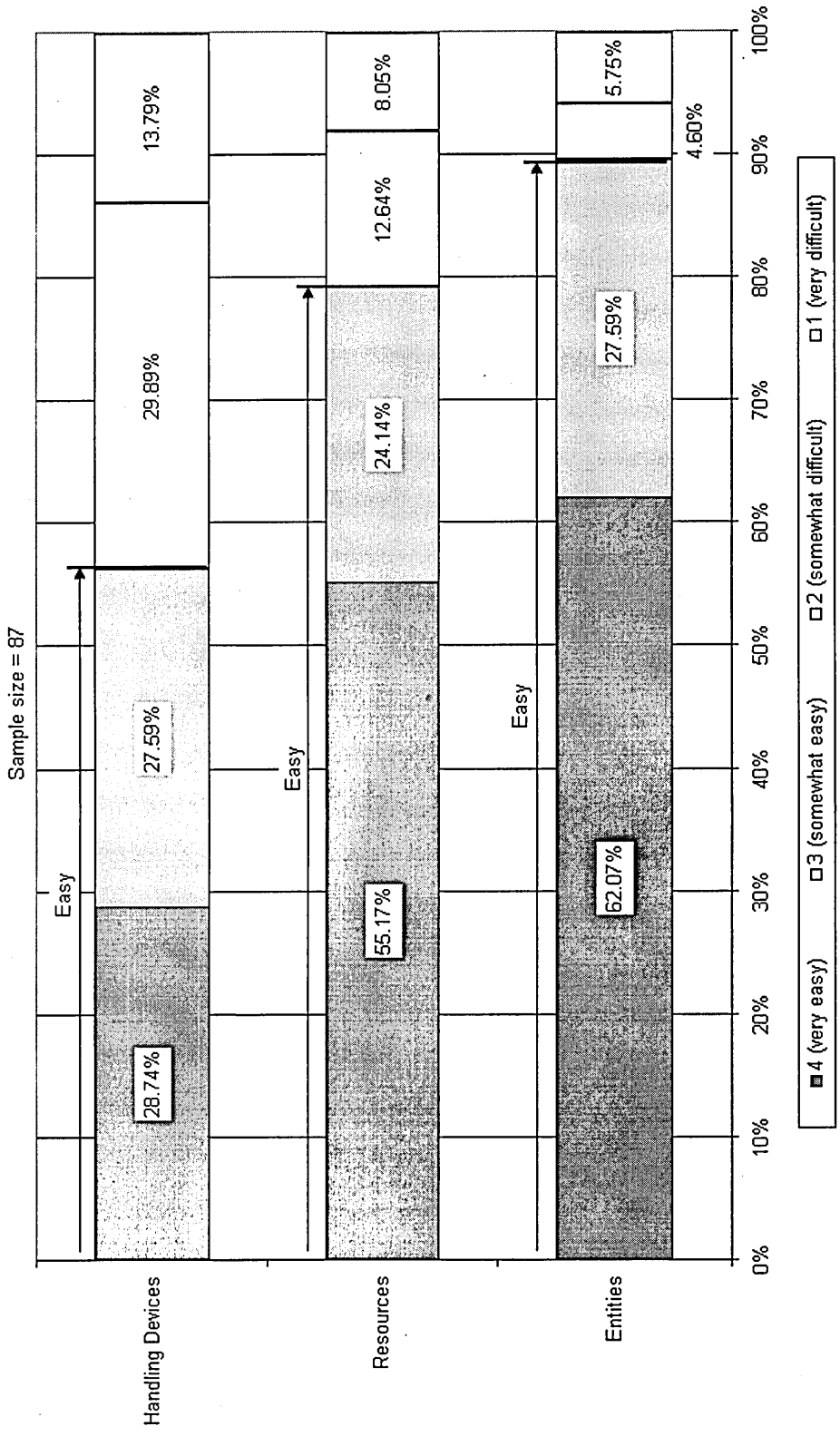

Figure 3: Difficulty in adding elements

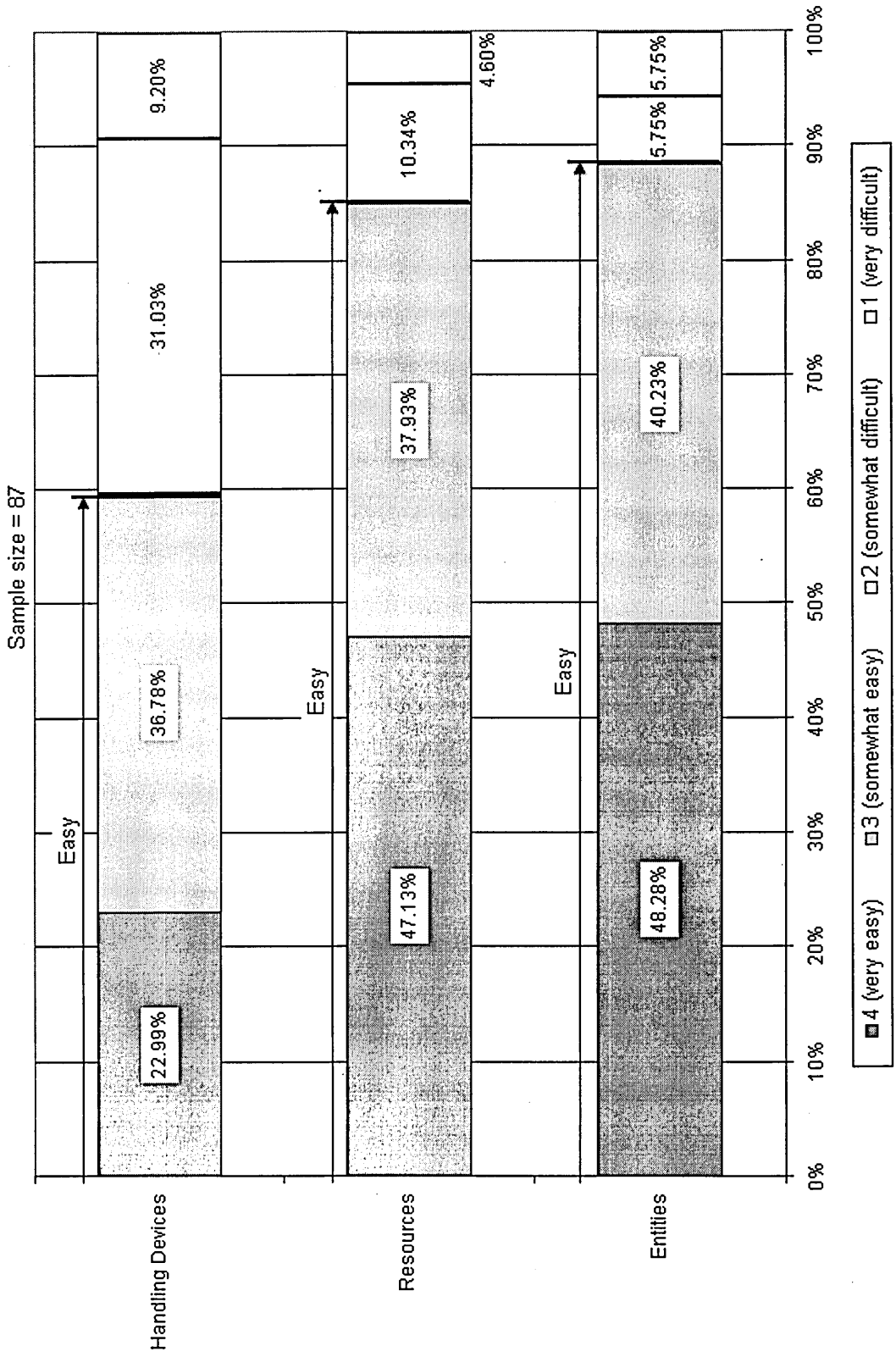

Figure 4: Difficulty in modifying elements

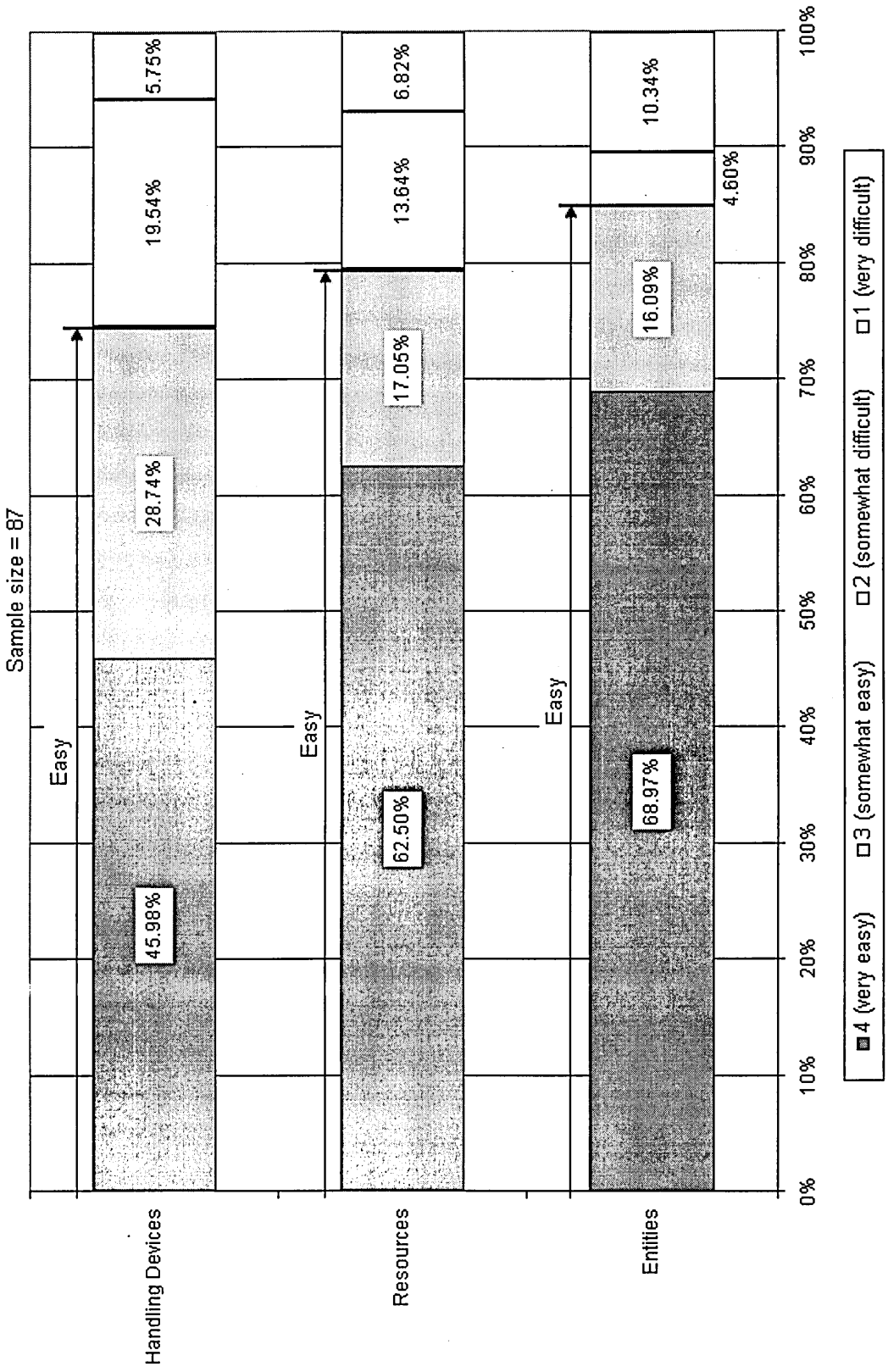

Figure 5: Difficulty in deleting elements

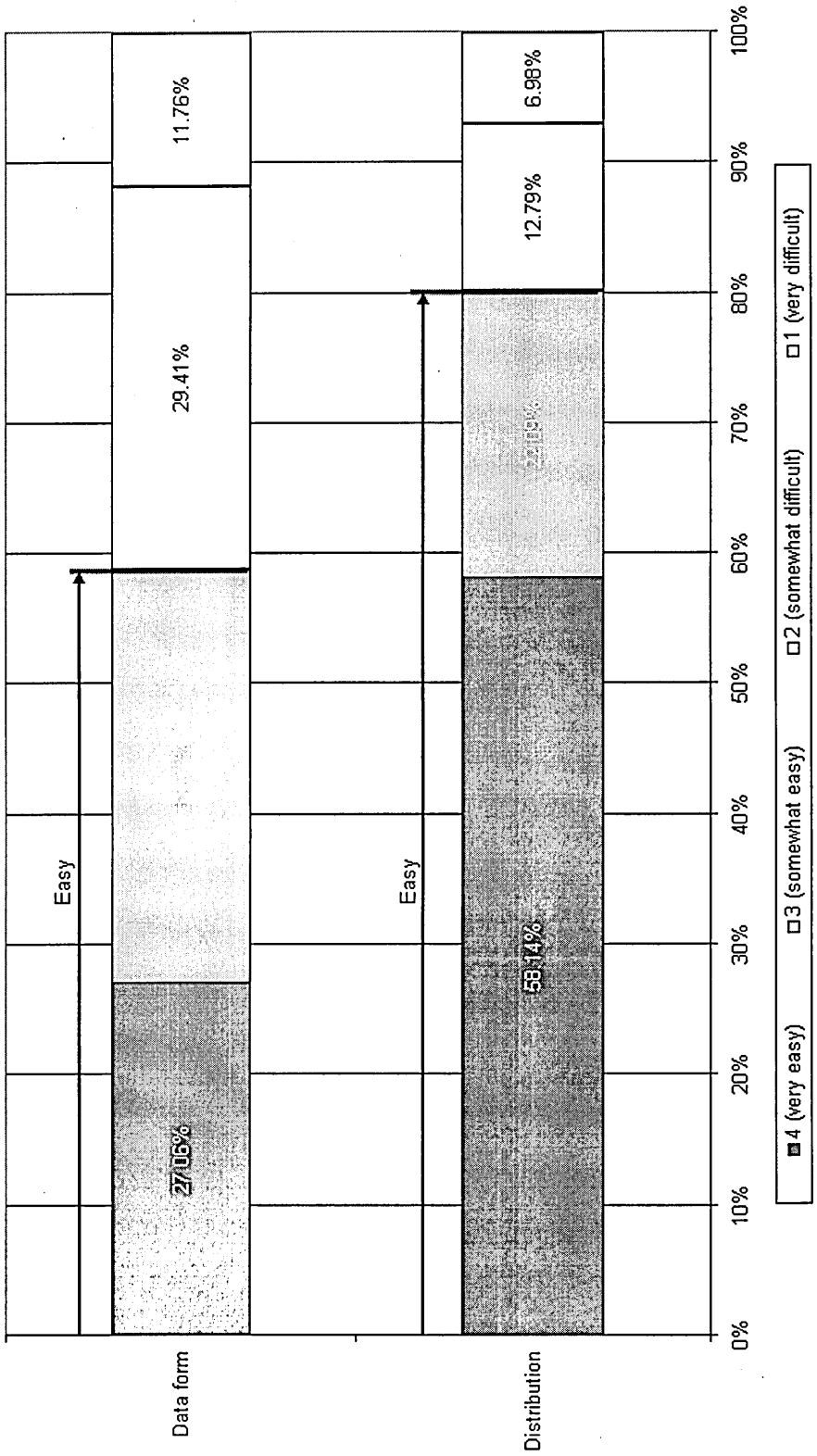

Figure 6: Difficulty in modifying distributions and changing form of data

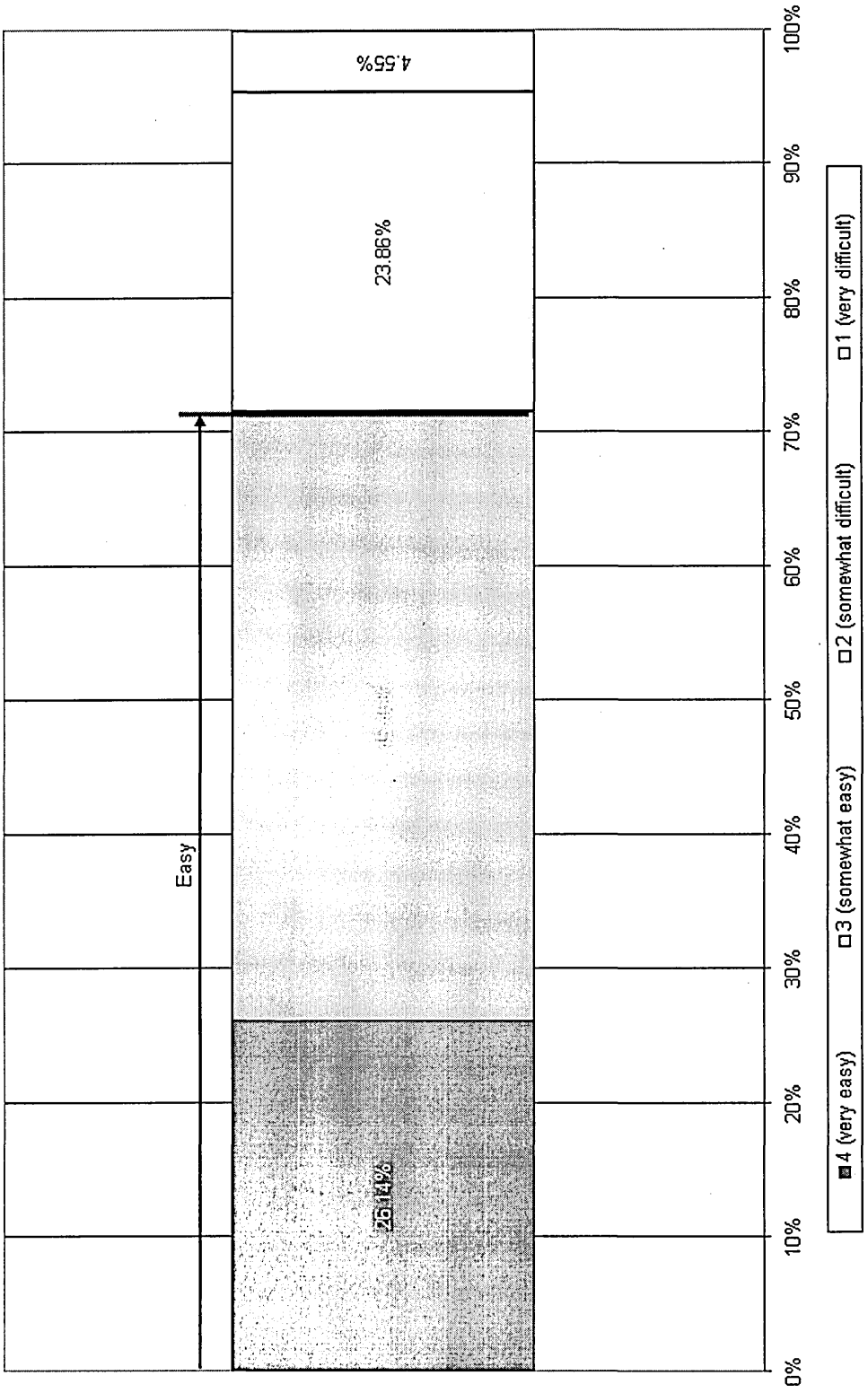

Figure 7: Difficulty in modifying animations

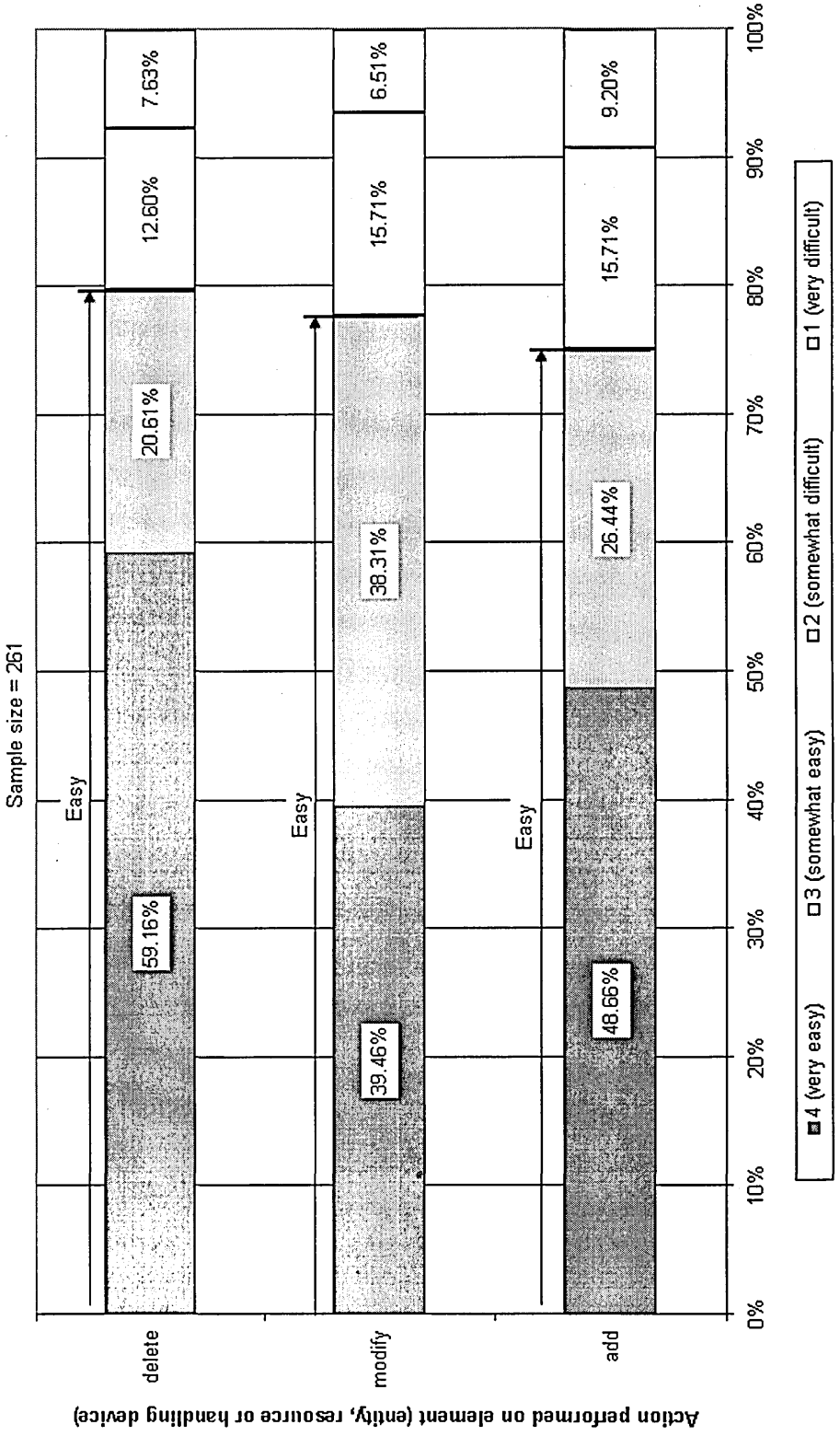

Figure 8: Difficulty in changing various elements

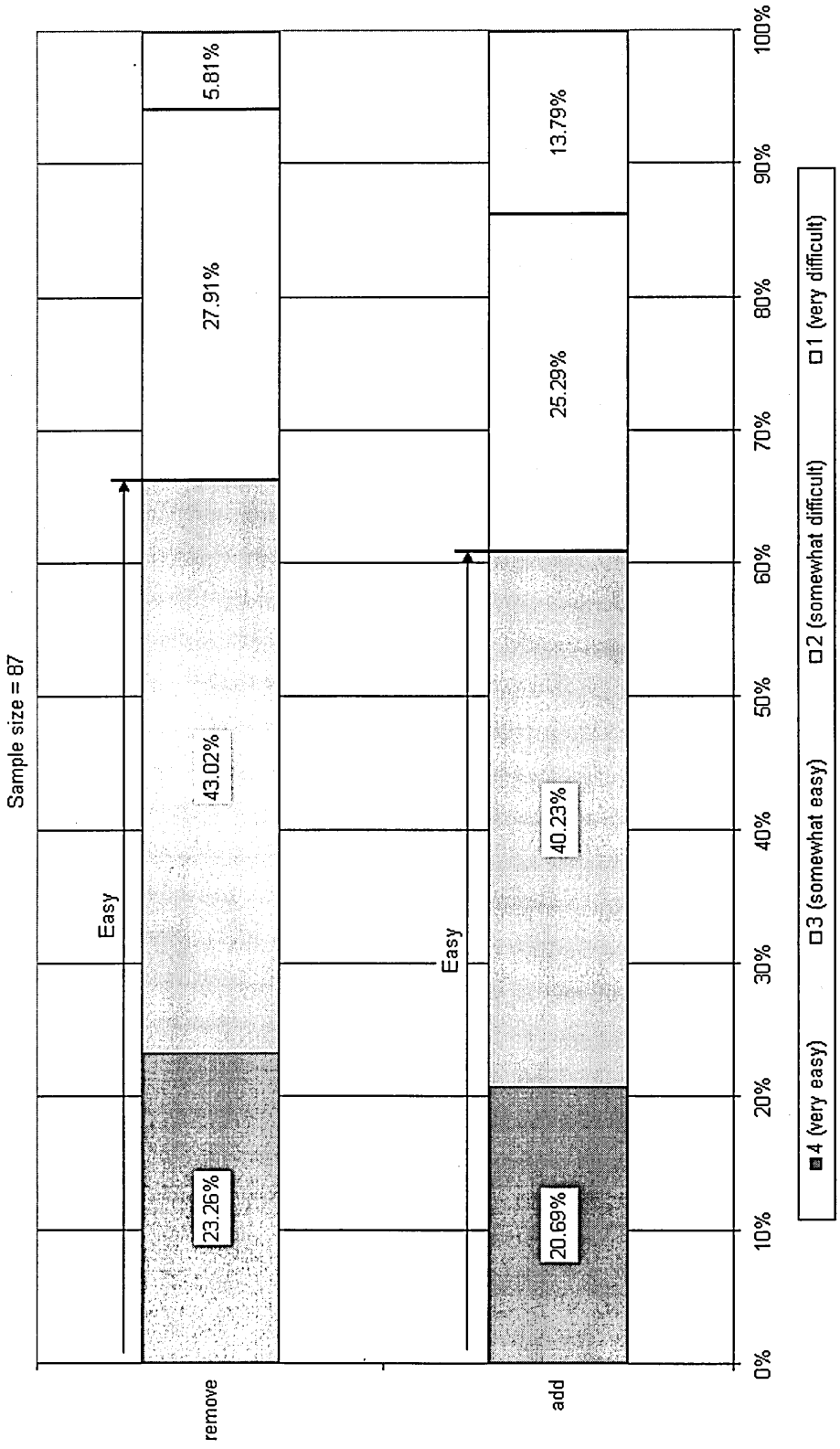

Figure 9: Difficulty in modifying model detail

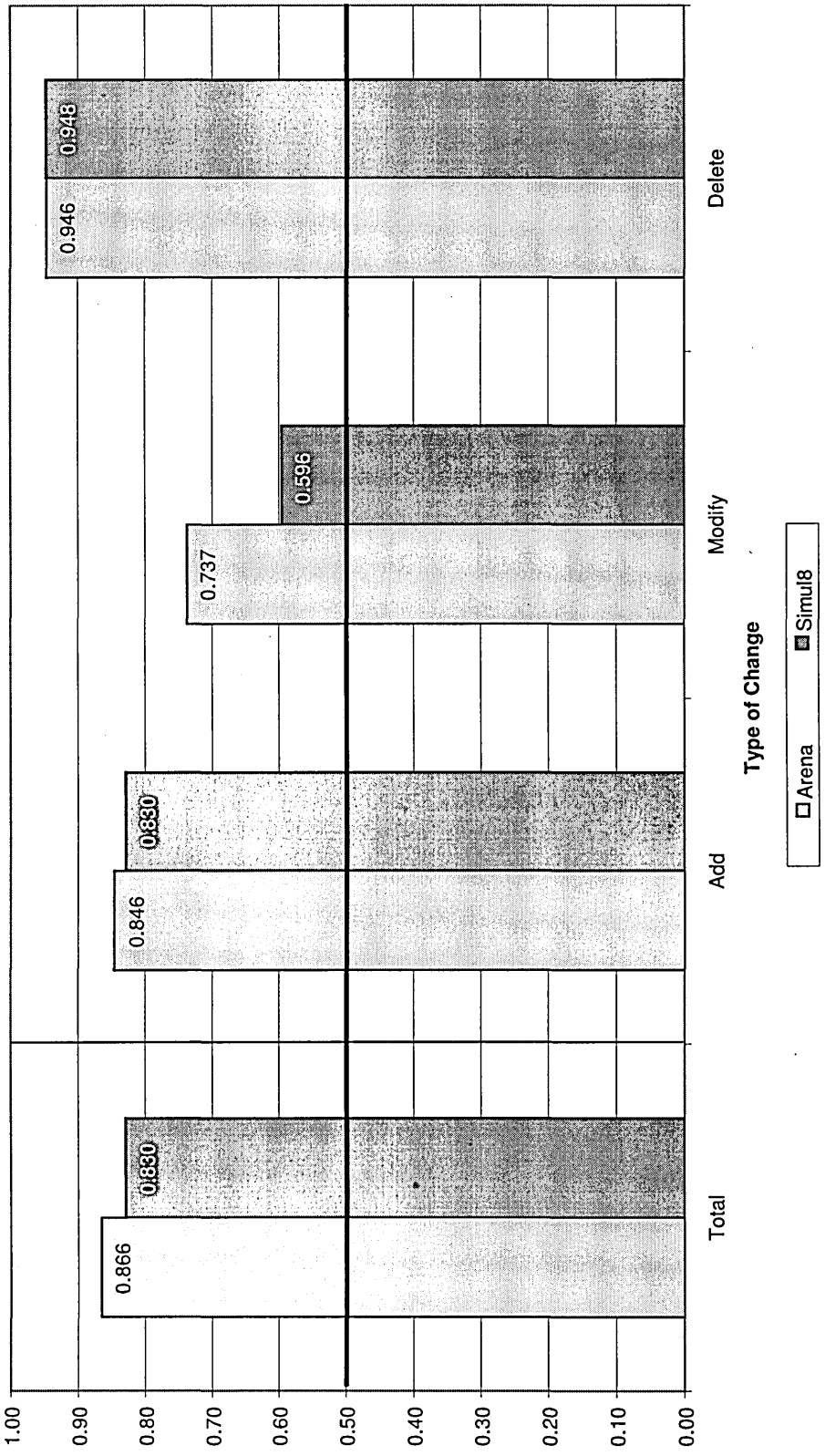

Figure 10: Adaptability without animation

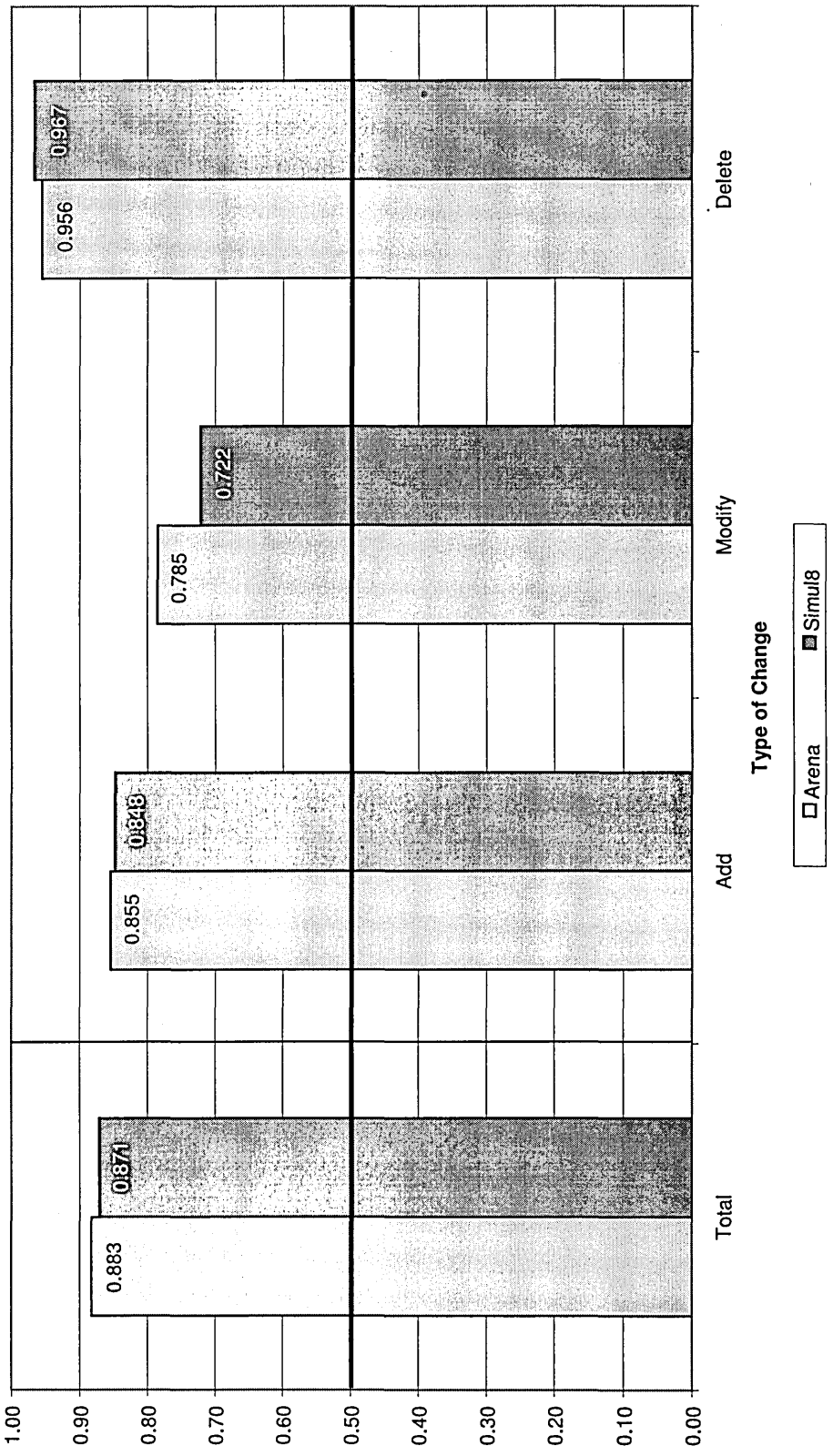

Figure 11: Adaptability with animation

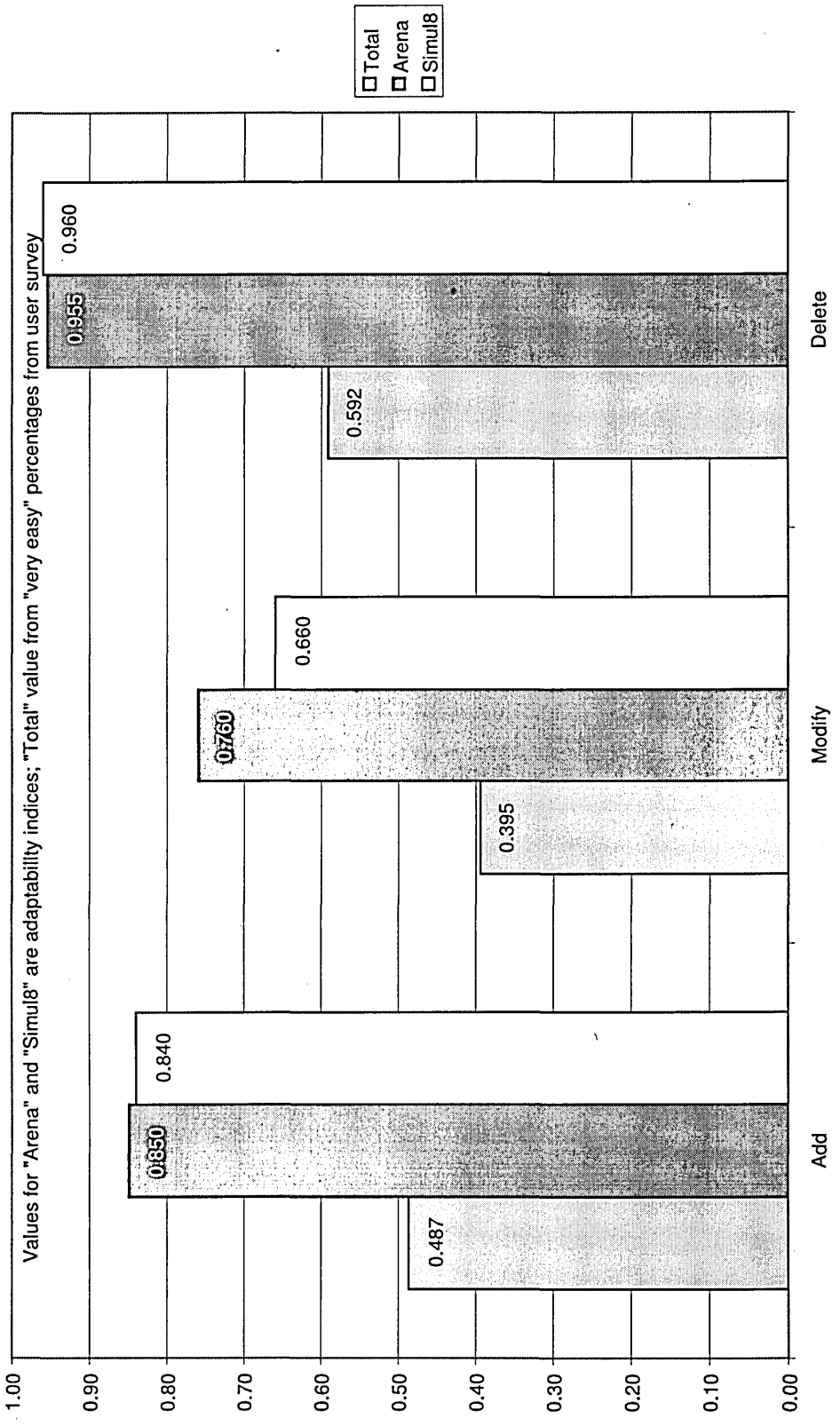

Figure 12: Adaptability (with & without animation averaged)

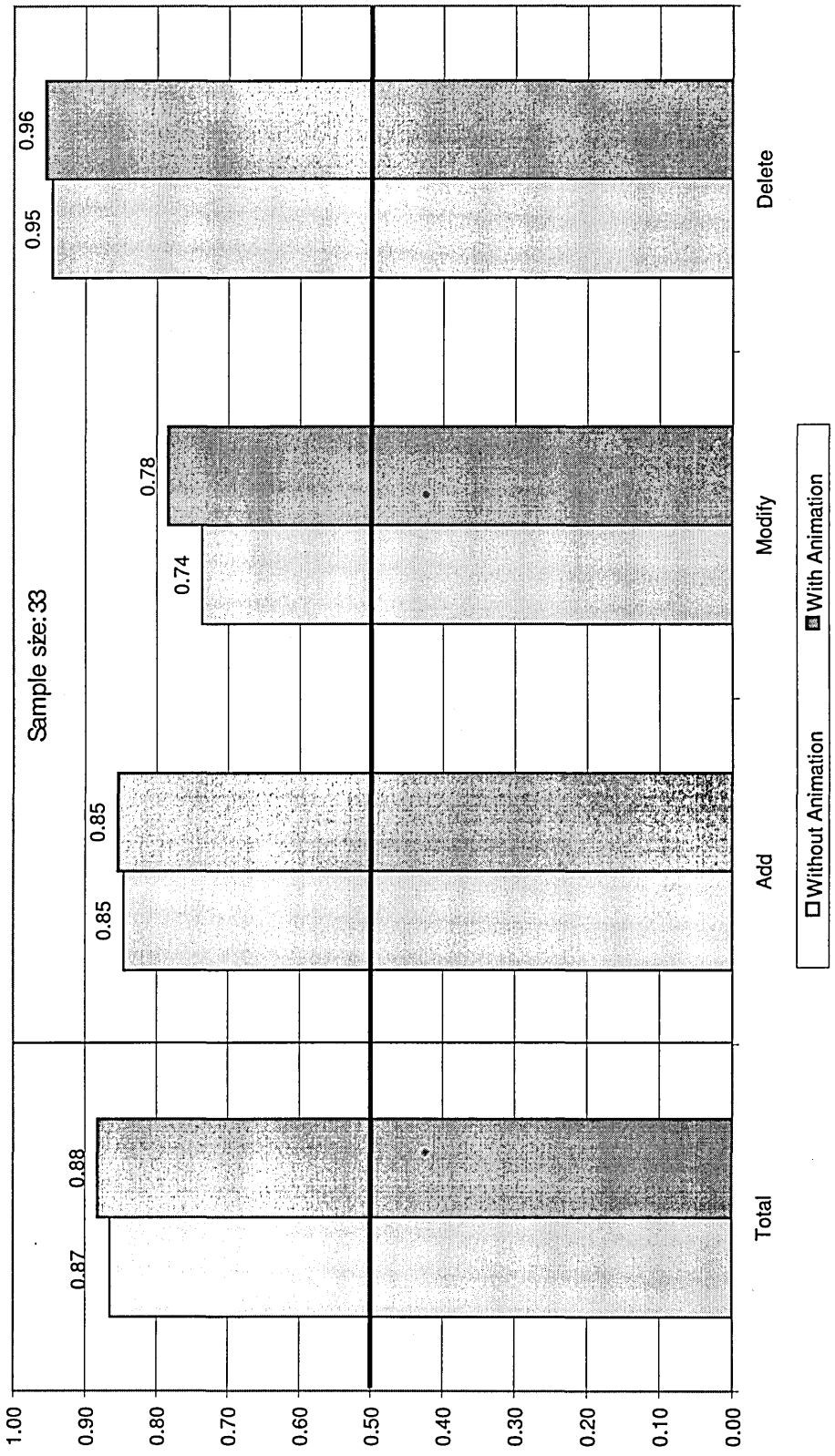

Figure 13: Adaptability of Arena

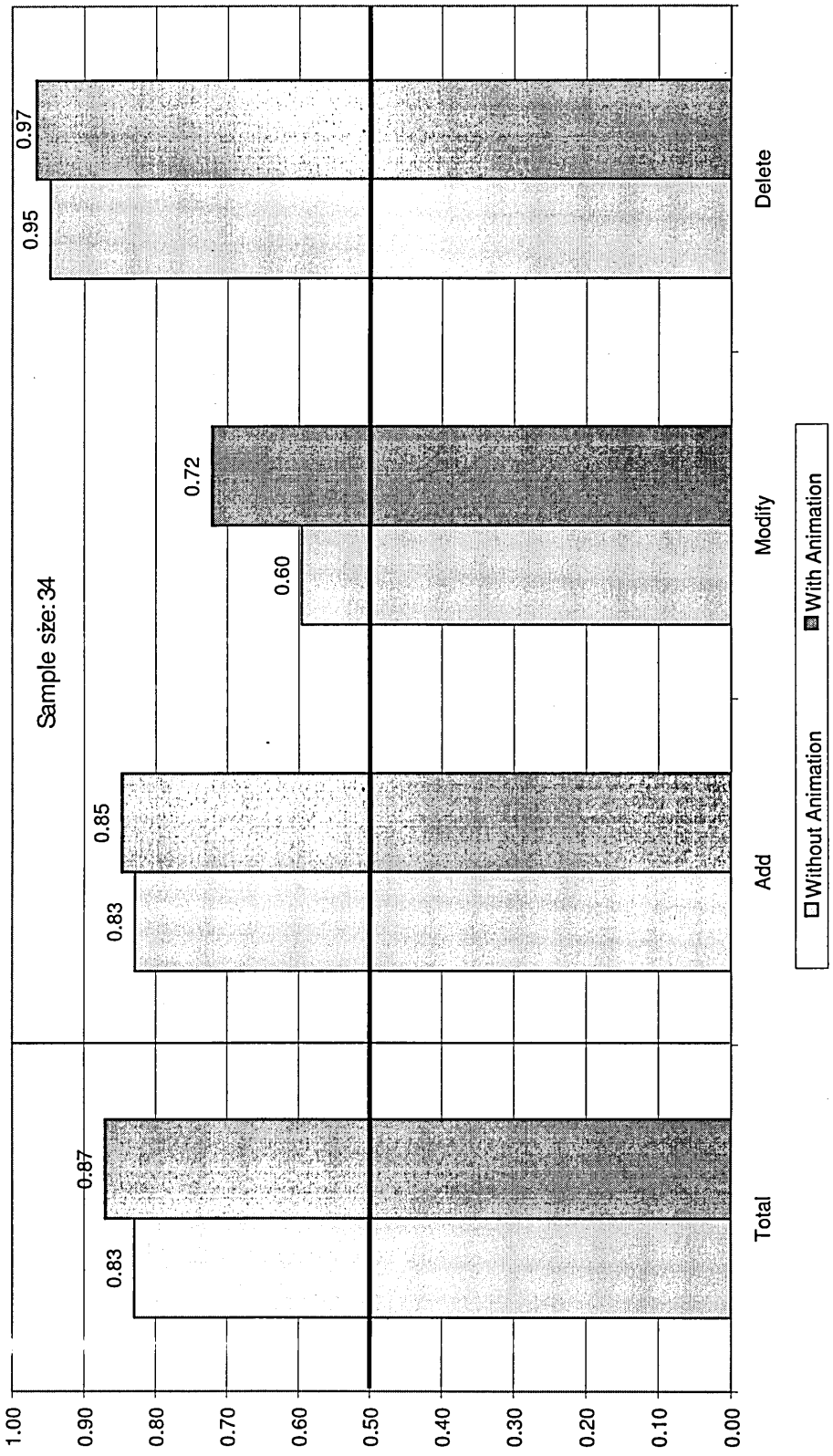

Figure 14: Adaptability of Simul8

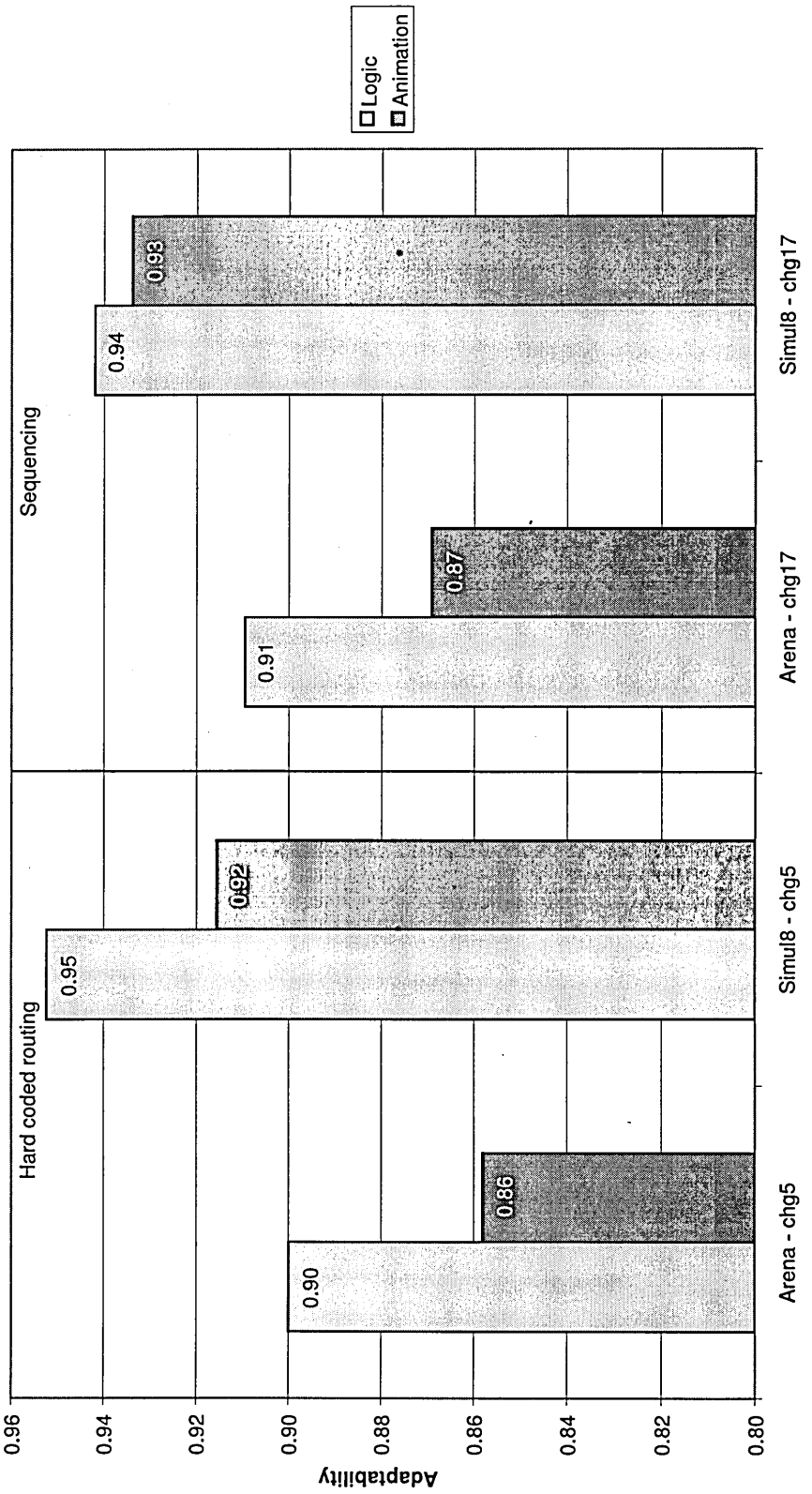

Figure 15: Influence of sequencing on adaptability

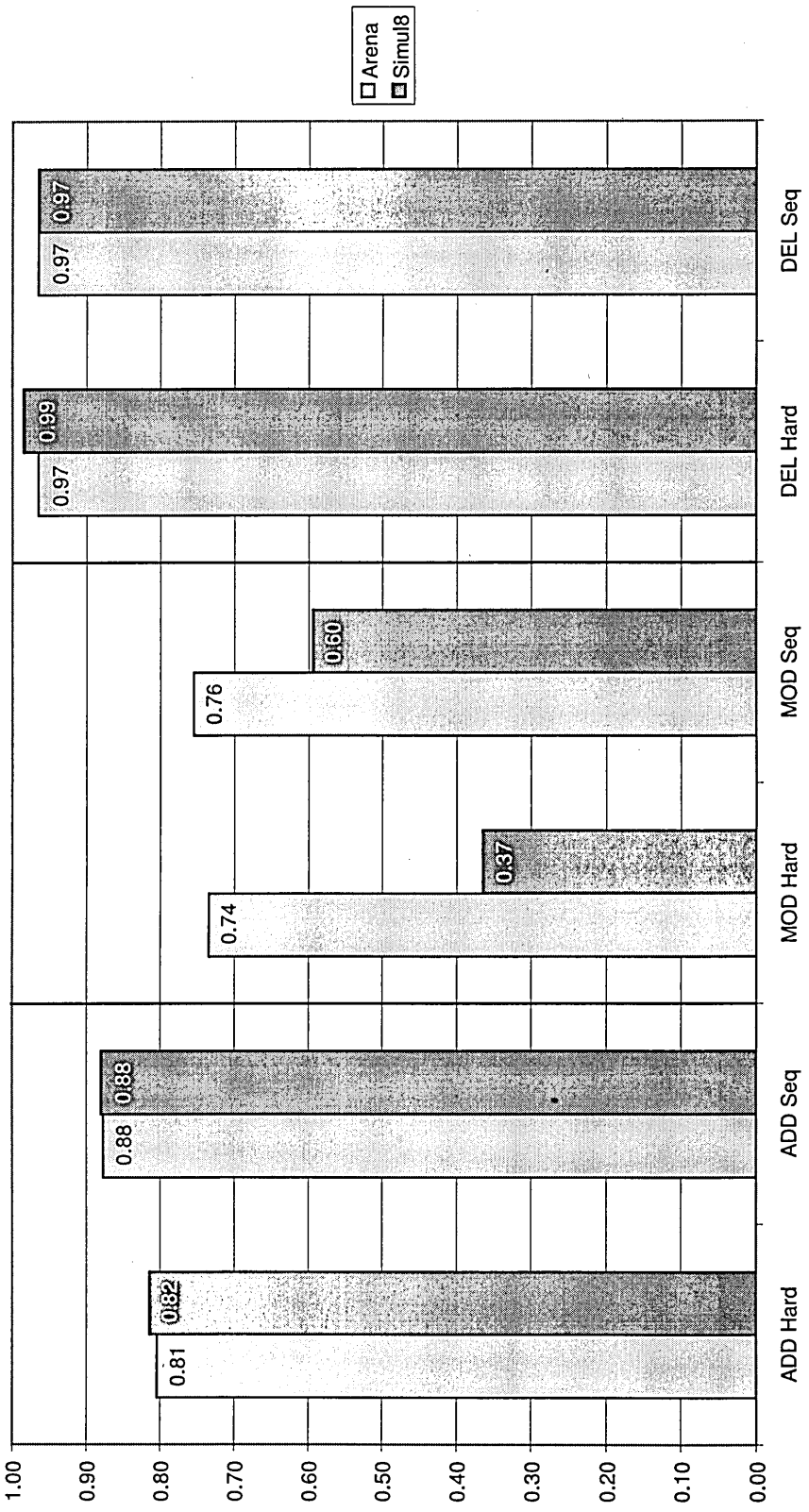

Figure 16: Adaptability of hard coding and sequencing by category

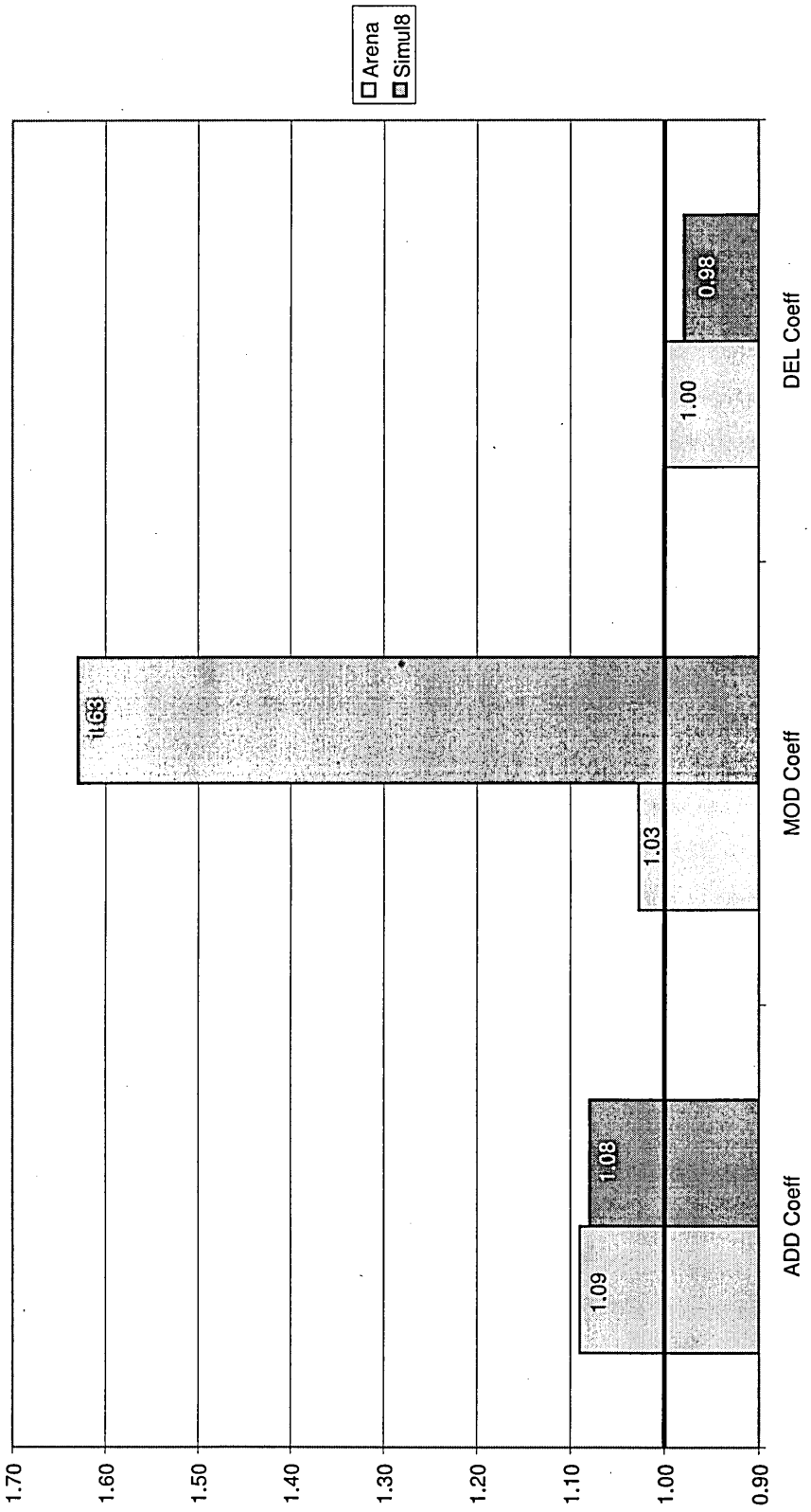

Figure 17: Adaptability coefficients

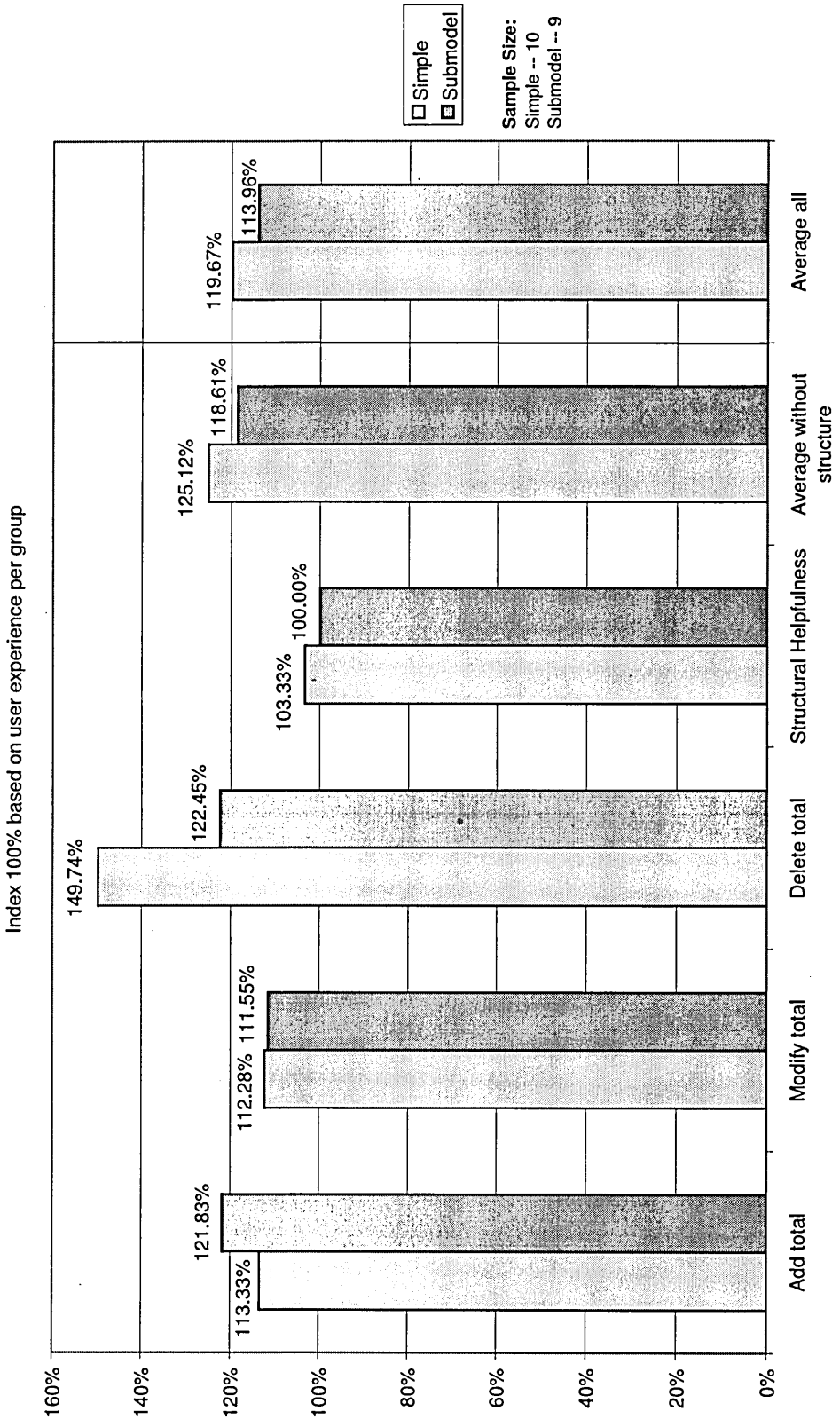

Figure 18: Influence of use of submodels on overall ease of use

Chapter 7 –
Validation of the Guidelines

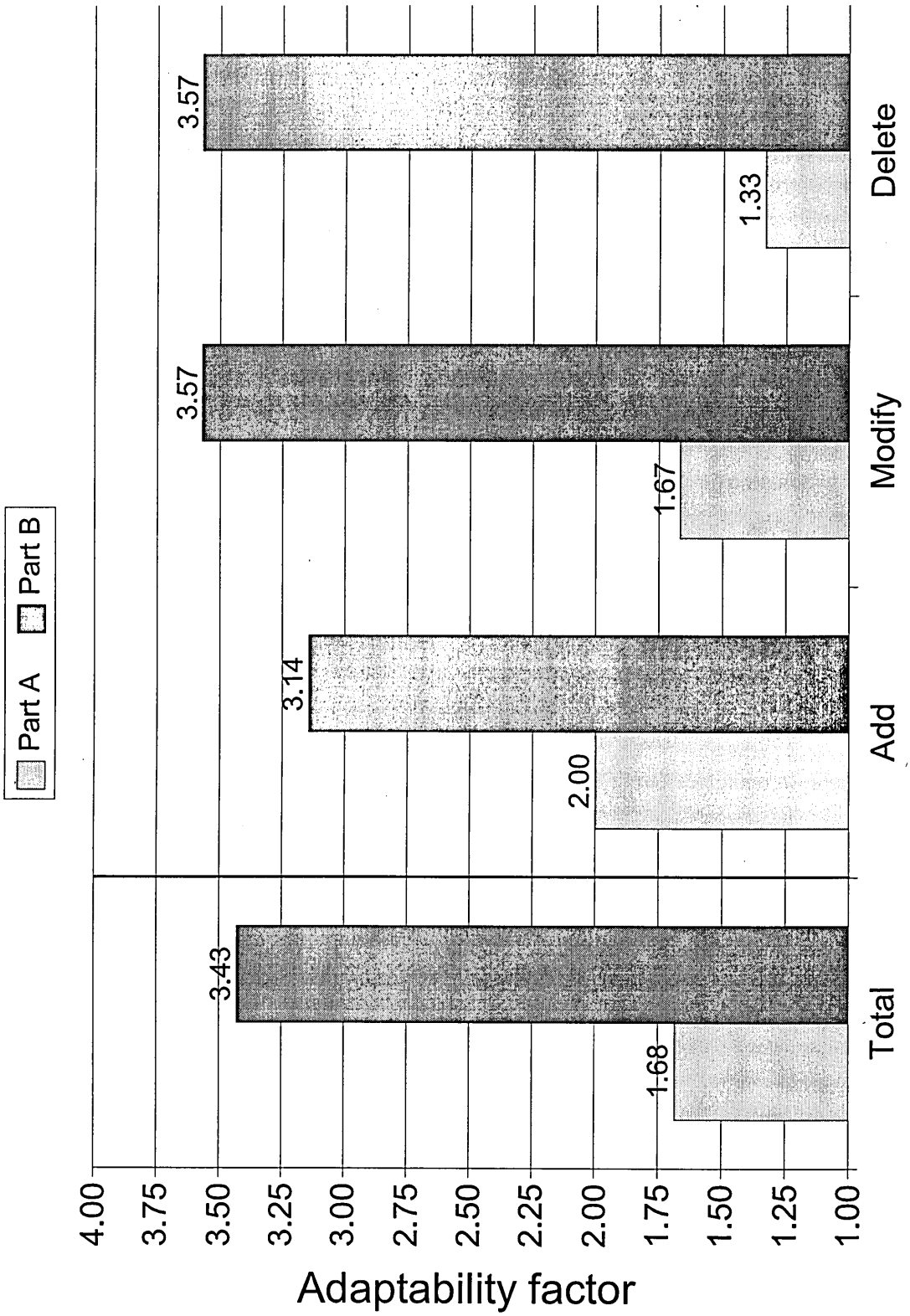

Figure 19: Adaptability factors