

Supporting Model Building

Master Thesis

by

Roland Marcus Servaes Groen

Supervised

by

Dr. Bert Bredeweg

June 2003

Social Science Informatics (SWI)

University of Amsterdam (UvA)

Roetersstraat 15 1018 WB Amsterdam

Contents

Abstract	v
Preface	vii
1 Introduction	1
1.1 Constructionism	1
1.2 Qualitative Reasoning	1
1.3 Educational contexts	1
1.4 Structure of the thesis	2
2 Theory	3
2.1 Introduction: a Theoretical Basis	3
2.2 Learning by Constructing	3
2.3 External Representations	4
2.4 Graphics	9
2.5 Support of External Representations	10
2.6 Qualitative Simulation and Modeling	13
3 High Level Tools for Building Models	19
3.1 Introduction	19
3.2 General Architecture	19
3.3 MOBUM	21
3.4 The SWAN Sketchpad	22
3.5 The Causal Model Pad	27
3.6 Evaluation	29
4 GarpApplet: The Output	33
4.1 Introduction: a GARP Simulation	33
4.2 Simulation Control	33
4.3 The State Graph	34
4.4 The Causal Model	41
5 Support	51
5.1 Introduction: Types of Support	51
5.2 Inherent Support	51
5.3 Avatars	53
5.4 Static Support	54
5.5 Dynamic Support	55
5.6 Intelligent Agents	61
6 The Experiment	65
6.1 The Research Question	65
6.2 The Assignment	65
6.3 The Method	65
6.4 The Means	66
6.5 The Prediction	68

6.6	Results of the Subjects' Evaluation	69
6.7	Results of the Subjects' Productivity	74
6.8	Conclusion	75
7	Conclusion	77
7.1	Specification Components	77
7.2	Integrated Model Prediction	77
7.3	Support	78
7.4	Evaluation	78
7.5	Future Recommendations	79
A	Agents	81
A.1	Communication	81
A.2	Reasoning	83
A.3	The Complete Rule Base	89
B	Experiment	103
B.1	The Questionnaire	103
B.2	The Experimental Data	113
B.3	The Details of the Statistical Analysis	119
	List of Figures	131
	List of Tables	135
	Index	136
	Bibliography	139

Abstract

Constructionism argues that learners should learn by articulating their ideas instead of being tutored. By building models, learners acquire knowledge. The articulation of knowledge forces learners to actively express, test, reflect upon, and adjust their conceptions. Qualitative reasoning (QR) is a language that provides computational accounts of human reasoning about the physical world, ranging from common sense ideas to the sophisticated knowledge of scientists and engineers.

This thesis is about how to create modelling environments based on QR technology that are suited for educational settings. In order to deploy QR in educational contexts, the existing software needs to be optimised for this purpose. Three extensions are proposed. First, *specification components* allow for model descriptions on a different level than the model building formalism itself. These components provide integrated sketching, behaviour envisioning, and early causal model articulation. Second, *integrated simulation inspection* is part of the model building process. A simulation inspection tool, capable of simulating and visualising the result of the simulation, is integrated into the model building environment. Finally, the modelling environment is equipped with different levels of support that scaffold the model building process. Intelligent support is provided to aid the modeler by providing reflections to the model, and communicating knowledge among the different components in the modelling environment.

An experimental evaluation and comparison of the newly created modelling environment and its predecessor shows significant results in that model builders evaluate the new environment more positively. Furthermore, the new environment seems to be more efficient.

Preface

I would like to thank all those people who, in some way, have contributed to this master thesis.

First of all I would like to thank my tutor, dr. Bert Bredeweg for his patience, time, good advice, and knowing how to temper my stubbornness. His enthusiasm on the subject of qualitative reasoning inspires both me and many other students.

I would like to thank my parents for their unlimited trust, motivation, support, and patience. Soon all their offspring will have a university degree. This is, without a doubt, a result of their effort.

I would like to thank the faculty of SWI for allowing me to use their facilities and embracing me as one of their own. I would also like to thank all those people who I got to know better at the SWI faculty. Special thanks goes to Vania for her friendship, presence, advice, help, support and, last but not least, showing me Brazil. Thanks for that! I would like to thank Machiel Jansen for the numerous conversations, his unique sense of humor and his advice on many issues. Gaston Heimeriks showed us all the importance of laughter, especially in the stressful moments. I would like to thank Gaston for his friendship, his patience as a travelling companion in Brazil, his humor, his extraordinary taste for music, and the many Friday-night beers. I would like to thank Anders Bouwer for his friendship, his work as the chairman of the Spekkie Big Fanclub, and his advice on writing. I would like to thank Gea Lindeboom from EPOS for her joyful presence and for the unlimited amount coffee. I would like to thank Noor for her pleasant presence and her advice on statistics.

Furthermore, I would like to thank the people close to me: I would like to thank Irina for her patience as girlfriend and as a friend, and of course for her preliminary reading. I would like to thank Guido Meijnders for being "al mijn vriend" in those many moments of social isolation and despair. Also thanks for his preliminary reading. Maarten van Hoof for his 24-7 GKOM help desk service, his support as a friend, and his numerous ideas of qualitative reasoning.

I would like to thank all the people from "De Jeugd van Tegenwoordig", Jaques, Barabra, Rintse and Maarten for their engagement on the subject of qualitative reasoning and education, and how to achieve complete world domination with this.

I will never forget the 8 years of Joan's joyful greetings.

Introduction

1.1 Constructionism

The idea of *constructionism*, advocated by Papert [47], argues that learners must learn by articulating their ideas. This opposes the idea of *instructionism* in which learners are considered to be passive actors who absorb knowledge by being instructed by a tutor. Constructionism assumes that by modelling, just as creating a soap sculpture in art class, learners acquire knowledge. The articulation of knowledge in models forces the learners to actively express, test, reflect upon, and adjust their conceptions of a domain.

1.2 Qualitative Reasoning

Qualitative reasoning is a language, or formalism, that provides computational accounts of human reasoning about the physical world, ranging from common sense ideas to the sophisticated knowledge of scientists and engineers [29]. People tend to solve problems by analysing the causes and effects in a problem domain. Qualitative reasoning enables humans to express their knowledge in a human-like way [29].

The computational nature of qualitative reasoning has a number of advantages. First, it facilitates knowledge communication. Qualitative models can be exchanged among teachers and students, supporting *collaborative learning*. Second, qualitative models can be predicted, or *simulated*. Learners can simulate their models and thereby get feedback on their work. Third, the automated nature of qualitative reasoning makes things like *explanation generation* and *automated tutoring* possible [53, 12]. These techniques aid the learner in the educational process.

1.3 Educational contexts

In order to use qualitative reasoning in educational contexts, software is required that is tailored to this situation. The technology is available, such as GARP [14], VisiGarp [11], and HOMER [39], but this software is not optimised for educational purposes. There is a need for modelling environments that scaffold the learner in the model building task. This can be realised by extending model building environments with abstract model building aids, integrated simulation inspection tools, and intelligent support:

- There is a need for specification components that provide a certain level of abstraction. Modelling does not start at the most detailed level, just as an architect starts off by raw sketches before making detailed (technical) drawings.

How can specification components allow for loose specifications or model certain specific aspects of a model, while omitting others?

- Qualitative models allow for prediction. The result of this prediction, or simulation, must be made insightful in order to inspect the result. Tools are needed for this purpose. Moreover, model inspection struggles with the problem of complexity. A large model can result in a very complex simulation, which is hard to conceive.

How can model prediction and inspection be integrated into the modelling environment?

- There needs to be a certain kind of support. The system is aware of the model under construction, and is able to reflect on it. Intelligent agents can give advice on the model in construction. Intelligent agents can aid the learner/modeler in the model building task by giving advice and providing facilities for communicating knowledge.

How can (intelligent) support be deployed in modelling environments to scaffold the modelling building process?

1.4 Structure of the thesis

This thesis is about how to provide these three extensions to the available technology. In chapter 2 the theoretical framework is discussed. This chapter goes into detail on subjects such as learning by model building, how external representation can facilitate this, how graphics can be deployed in model building environments, how to provide (intelligent) support, and why we should prefer qualitative modelling over quantitative modelling. Chapter 3 discusses two specification components. Specification components provide levels of abstraction in the model building process. One tool is a sketching environment that provides the modeler with a high level of freedom and meanwhile facilitates the first step of model refinement. The second tool enables the modeler to express a causal model of their conceptions in an early stage of the modelling process. This tool can act as a causal test bed. In chapter 4 a model inspection tool is discussed. This tool provides means to inspect the result of a simulation. The emphasis of this chapter is in order to employ visualisation strategies on how to make a simulation insightful to a model builder. Chapter 5 focusses on how to provide support. First, the user interface can anticipate on the task the user has to perform and attempt to be ‘self explanatory’, that is, easy to use. Second, a hypermedia help system can provide “static” help. Finally, intelligent agents can provide reflections and support based on the current status of the model. Chapter 6 describes an experimental evaluation comparing two model building environments. The two environments differ in the level of support they provide. The experiment shows that the tool equipped with (intelligent) support is evaluated more positive. This suggests that there is evidence that (intelligent) support does aid learners in their model building effort.

2.1 Introduction: a Theoretical Basis

This chapter discusses the theoretical basis for this thesis. The general question is how do we make educational tools that support model creation, modification and inspection. This question covers several research areas. First, the educational value of constructing models is discussed. The question of why we want learners to make models in the first place is addressed. Second, we discuss how graphical representations can help learners in their model building task. What kind of graphics are necessary to support building models? Third, we discuss what facets make a tool easy to use and how to provide support to the model builders. And finally, we discuss qualitative modelling.

2.2 Learning by Constructing

New technology has emerged in the past few decades. All parts of society have been subject to changes initiated by new technology. Education is no exception. Technological innovations have acted as a catalyst on new concepts in education, by providing new means of communication and interaction. One of the first tools written purely for educational purposes was Logo [47], a programming environment for children. Different tools and applications for educational purposes have emerged since.

The new technologies deliver new functionality, not yet known to education. This rises the question of how to deploy these new possibilities. What kind of tools do we need and how does this support learners in learning? Does new technology actually support learners better than traditional methods in education? These are the questions that research in AI and education¹ challenge.

One of the concepts used in new computer based tools for education is the concept of model building. Learners learn by creating, running and adjusting models in a model-building environment. Learning by building models is becoming a popular research area [28]. In the next section we address the question of how model building can facilitate learning.

2.2.1 Constructing Internal and External Representations

Jonassen [40] argues that experts are better problem solvers than novices because they construct richer, more integrated internal representations. Experts are better in classifying problem types because their representations integrate domain knowledge with problem types. It is generally accepted that problem solvers need to construct an internal representation to solve a problem.

Problem solving must begin with the conversion of the problem statement into a representation. Learners must connect the problem to domain understanding. In order to do so, they must relate the original problem to a meaningful representation that can be manipulated. People tend to represent problems in ways that fit their problem solving needs. For instance, spatial mappings are used to solve time scheduling problems.

¹The International Artificial Intelligence in Education (AI-ED) Society is concerned with this research area. They can be reached at: <http://cbl.leeds.ac.uk/ijaied/aiedsoc.html>

Jonassen [40] argues that there are qualitative and quantitative means of representation in learning how to solve problems. The quantitative method in education is: plugging in the right algorithm for the phenomena to solve the (mathematical) problem. By doing this, the right answer just pops up. But this is of little relevance to the conceptual understanding of the problem.

In contrast, experts use qualitative representations of the problem before finding a quantitative solution. It is important to note that qualitative representations support the solutions of quantitative problems. Training learners to qualitatively represent a problem improves learners' problem solving performance.

Providing learners with tools to model different kinds of qualitative representations will affect their ways of representing problems in order to scaffold their representation performance. Tools become part of the cognitive apparatus of the learners. When learners internalise tools they begin to think in terms of the tools' concepts and structure.

2.2.2 Constructionist Learning Paradigm

The concept of *constructionism* advocated by Papert [47] advances the claim that a way to build good internal representations is to produce physical or computational constructs which can be manipulated and modified by the learner. This contradicts the classical way of teaching: instructionism, in which learners do not learn by actively constructing, but by studying text books. Papert is interested in computational tools for math education. Inspired by the way learners work at soap-sculptures in art class, Papert advocates the idea of learning-by-making.

Wilensky [62] applies these principles in the theory of *connected mathematics* and responds to the view that mathematics must be seen as "received" or "given". A connected mathematics learning environment focuses on learner-owned investigative activities followed by reflection. Mathematical concepts are multiple represented and the focus is on learners building their own representations. Learners are supported in developing their own mathematical intuitions and building concrete relationships with mathematical objects. The idea that mathematics is not just received (and formal) implies that the domain must not be simply animated by the use of technology, but instead provided as a medium for the design of models by learners.

2.3 External Representations

When solving a complex task, people tend to use artifacts, or external representations, to help them [18]. When making a complex calculation, paper and pencil come in handy to store an intermediate result. When going for groceries, a grocery-list helps us to remind what we need to buy. When travelling, a map helps us to find the way and highlighting the destination on the map will help us to remember the destination. These are examples of how humans use *external representations*.

In the previous section we have argued that students should build external representations for various reasons. Given that we want to make students modelers, how do we do this? External representations are powerful means for thinking and learning.

2.3.1 Definition of External Representations

Zhang [64] defines external representations as:

"...the knowledge and structure in the environment, as physical symbols, objects, or dimensions (e.g. written symbols, beads of abacuses, dimensions of a graph, etc.), and external rules, constraints, or relations embedded in physical configurations (e.g., spatial relations of written digits, spatial layouts of diagrams, physical constraints in abacuses etc.).", Zhang [64].

An external representation lays outside the person and plays a role in the process of human cognition.

2.3.2 How do External Representations Work?

The question of how external representation work can be analysed using three characteristics of external representations, identified by Scaife [54]. Before discussing the underlying theories, these three characteristics are clarified:

- First, there is the concept of *computational offloading*, reducing the cognitive effort required to solve problems. This can be done by representing the problem in such a manner, that it requires less effort to store, retrieve and search information in the problem domain. Larkin and Simon [43] argue that explicitly representing the problem state in diagrams enables solutions to be more readily ‘read-off’. External representations influence the *amount* of information we process in our reasoning task.
- Secondly, closely interwoven with the previous concept, is the concept of *re-representation*. This refers to how different external representations that have the same abstract underlying concept, can influence problem solving. Zhang and Norman [46] argue that the nature of the representation allows different operations to the problem domain and thereby influences the process of problem solving itself. External representations change the way we choose our operations to manipulate the problem. In other words, the *way* we reason is influenced.
- These previously mentioned concepts can find their cause in the concept of *graphical constraints*. Graphical elements in representations are able to constrain the kind of inferences that can be made about the underlying represented world. The mapping of the graphical objects on the features of the problem space restricts the kind of inferences that can be made in the problem space. Thereby simplifying the reasoning process to offload the computation or influencing the way the problems are solved by their re-representing features.

Zhang and Norman [64] [46] describe a framework for external representations. They consider internal and external representation as equal partners in the representational system. The information in external representations can be analysed and processed by the perceptual system alone, although the top-down participation of conceptual knowledge from an internal representation can sometimes facilitate or inhibit the perceptual process of interpreting the external representation. Zang explains how this works: representations facilitate or inhibit different *operations*, and not *visa versa*. External representations activate *perceptual operations*, such as comparing size and location of objects, making certain information directly accessible, without any inference. In contrary, internal representations activate *cognitive operations*, such as adding numbers to get the sum. When doing additions, internal representations have information that can be directly retrieved, such as the relative magnitudes of single-digit numbers.

The process of *lookahead* is the mental process of predicting the effect of a set of actions on the current problem state in order to resolve the problem. Think of a chess player planning a next move. This set of possible operations is too large to process, thus there must be a certain kind of choice, what operations to mentally investigate and what operations to ignore in the mental lookahead. First, the properties of external representations influence *what* operations are selected, they activate more *perceptual operations* and thereby *bias* the lookahead by biasing the selection of the operations. The influence of external representations on the *way* we reason, by biasing our selection of mental operations, is referred to as the phenomena of *re-representing*. The different ways of representing as opposed by the external representations influences our way of reasoning. Second, external representations influence *how much* operations are evaluated, since they have an influence on the computational effort required by the lookahead. External representations make the computational effort more tractable. This is the phenomena of *computational offloading*.

Zhang and Norman explain that external representations influence the selection of operations by their properties. Stenning and Oberlander [57] specifically address *how* the properties of graphical external representations influence our reasoning. External representations can change the nature of our reasoning process in such manner that the reasoning process become more computational feasible. They argue that graphical representations vary in their mappings to the things they represent. They believe that representations that are stricter in their mappings, that is, each symbol is limited to a certain concept instead of multiple concepts, behave differently from representations that are loosely coupled with their underlying concepts.

The level of '*specificity*' in representation limits the abstractions that can be made. This should lower the computational effort needed for reasoning.

Stenning and Lemon [56] support this view on strictness. They argue that the *availability of constraints* extend the way diagrams are interpreted. Constraints on the expressiveness are available to the user who has only a simplified understanding of their core semantics, whereas sentential systems provide no clues in their representational and inferential capabilities. The constraints on the represented relations must match those of their corresponding graphical or spatial relations. Those constraints must be available to the user.

Cox and Brna [18] discuss that the architecture of the working memory may make visual representations more feasible than their textual counterparts. Working memory exists out of multiple components. The central processing component, or *central executive* is connected to two subsystems. First, the central executive is connected to an auditory component, or *auditory channel*. Second, the central executive is connected to a visual component, or *visual-spatial sketchpad*. The auditory and visual components differ in nature on how they represent and access information. Spatial information is encoded automatically and independent of attention into the visual spatial sketchpad, while auditory encoding needs active processing from the central executive. Visual-spatial encoded information can lower the workload on the central executive because visual-spatial information is encoded automatically. Bringing in the visual-spatial sketch-pad for representing information by representing the problem in a spatial manner, will make the inferences about the underlying domain more feasible.

Larkin [43] argues that graphics serve to illustrate structure. The visual form constrain the ways in which representations can be used. Errors in processing often result from invisibility of function and the non-salience of form. External representation need to be well constructed and represent the information in the problem. If both criteria are met, the rapid processing capabilities of the visual system can be exploited. Very easy perceptual judgements are substituted for more difficult logical ones.

2.3.3 Multiple External Representations

Representations tend to be plural, unstable and evolving. People seldom represent problems with a single representation, instead people use several representational systems. Furthermore, the idea is that rich, overlapping collections of different views and considerations is much more a characteristic of human knowledge than a single representation, and thus, are beneficial when solving problems [24].

Petre et al.[48] discuss the cognitive aspects of multiple representations. They argue that multiple representations are not necessarily better. They evaluate multiple representations in the context of programming and programming environments. A single representation uses less screen space, avoids problems of switching from one to another and finding the right place in each one. Furthermore, it avoids problems of confusing which part from one representation belongs to which part of the other. Petre et al. discuss reasons for programmers to use different representations. In the most simple case, programmers use different representations for different types of tasks. *Identical representations* are used to simultaneously have different views on the same matter, think of architects using different perspectives to visualise a 3D building. The types of representations are the same, their view differs. *Bridging representations* facilitate search and reasoning strategies. A specific representation may facilitate a specific search strategy, while other forces exploration or facilitate indexing. *Heterogeneous inference* is a case of multiple representations where two forms of representation need to be used simultaneously for best results. Labeke [60] did research on multiple representations, and used different views to support students in understanding Euclidean Geometry. In a task of constructing proof out of a geometric world, two different representations are used: one logical, textual, and one graphical representation. The two representations are complementary, graphical input stimulates perceptual operations, textual input stimulates cognitive operations.

External Representations in Education

In the context of education, Brna et al. [15] provide an overview on external representations. They argue that the self-explanation tendencies, found in students, can be assisted by multiple external representations. They found that effective students more frequently explain and justify actions of themselves, monitor their performance more accurately; and refer back to examples for specific pieces of information. In other words

their metacognition is more elaborated. Multiple representations might stimulate students in considering more than one model in their evaluation. However, there is evidence that students have difficulties coping with multiple representations. Simply providing coordinate representations does not guarantee success. Indeed, in certain circumstances multiple representations can 'act against' each other.

Ainsworth et al. [1] did empirical studies in the area of multiple representations. They summarise the benefits of multiple external representation:

- *Different ideas and processes.* Using different combinations of representations, we can exploit their different properties to aid learning. New representations can be used in combination with known representations to develop better understanding.
- *Constrain interpretations.* They clarify a situation by constraining learner interpretations. Constraining factors in representation can facilitate understanding of the domain.
- *Provide deeper understanding of the domain.* Multiple linked representations may allow learners to perceive complex ideas in a new way and apply them more effectively. In supporting abstraction, mapping between the representations is crucial.

Drawbacks of the use of multiple representations are that the learner first must come to understand each individual representation. Second, they must understand the relation between the different representation; and third, in certain circumstances, how they relate to each other when they are dynamically linked. These can be very demanding learning demands [61]. Ainsworth et al. conclude that the usage of multiple representations depends on the teachers goals. Knowing what a representation offers and how to use the representation is crucial. Successful abstracting depends on the trade of between the benefits and drawbacks of multiple representations.

2.3.4 Sketching

As mentioned before, external representations can be very useful in educational contexts. The constraining and highlighting features of external representations influence how we perceive and solve a problem. This can be productive, since it will force or guide the user to think in a certain way. The graphical syntax maps directly on the concepts in the represented modelling language the model builder is creating.

In the early phases of articulation, this restrictiveness in representations is unwanted. Architects, engineers, and designers make raw sketches of their ideas before becoming more explicit. To draw the analogy with architects: they use many levels of abstraction before coming to the final specifications. Sketches are used for communicating early ideas. Scale models are commonly used to communicate the design ideas to the client. Intermediate drawings are used for technical calculations and communication purposes.

Figure 2.1 depicts the process of model construction. The process iterates over creation, testing and reflecting. We argue that this process starts off with vague ideas and ends with a complete model. At the different levels of abstraction, different levels of constraints should apply. Ullman et al. [58] argue that sketches are essential during *all* stages of the design process and are necessary extensions of a designers cognitive capability.

Lipson and Shpitalni [44] point out the important aspects of sketching:

1. Sketching is *fast*, suitable for the capacity of short term memory.
2. Sketching is *implicit*, it describes a form without a particular sequential structure.
3. Sketching serves for *analysis*.
4. Sketching is *inexact* and *abstract*, avoiding unnecessary detail.
5. *Minimal commitment* is required, it is easy to discard and start anew.

Ullman et al. [58] and Forbus et al. [30] argue that sketching is interactive drawing plus linguistic interaction or textual connotation. The drawing carries spatial aspects of what is to be communicated. The linguistics provide a complementary linguistic channel that guides the interpretation of what is drawn.

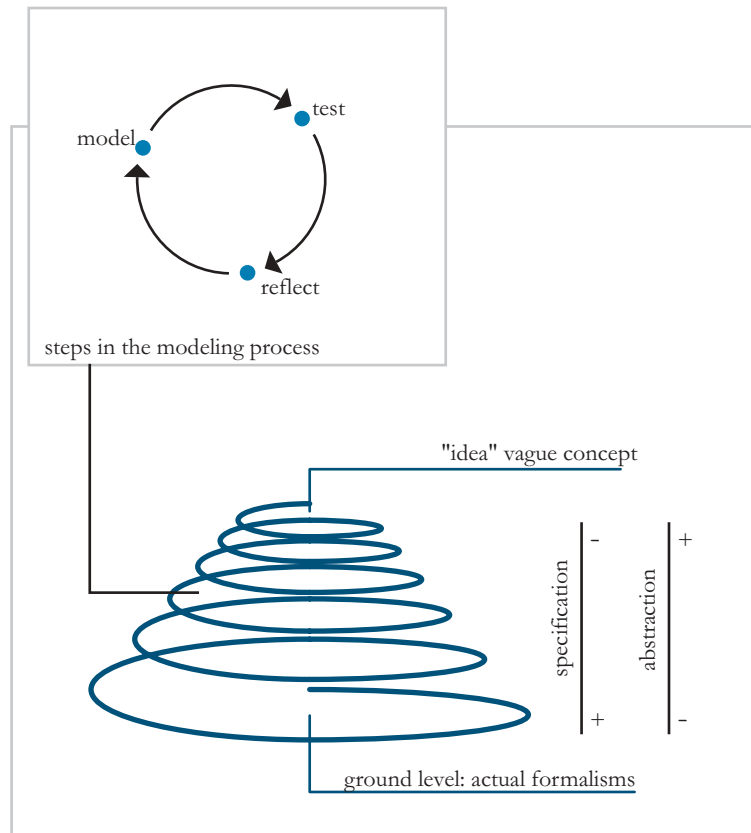


Figure 2.1

The process of model construction.

Why do we want to support sketching in automated environments? Alvarado [3] argues first, that the idea does not have to be expressed twice: once by sketching, and the second time by transferring it into the computer. Second, it would be easier to edit the sketch. Third, the process can be recorded. Finally, the computer can simulate the design. The latter is simulation of the physical behaviour. Other advantages can be thought of, such as importing library drawing, easy communication and exchange over great distance, etc. But the most important aspect, on the viewpoint of model building, is the ability to support various levels of abstraction by gradually allowing the user to mold their representation in a more strict graphical syntax.

Sketching tools can be divided into three groups:

1. Tools that provide sketching interfaces to a specific graphical syntax, transforming the sketching into that syntax, such as ASSIST for CAD, [2] and Tahuti [35] for UML.
2. Tools that leave the sketches intact, but attempt to infer information from them, such as GeoRep [25] and sKEA [31] frameworks.
3. Tools that uses sketches to connotate existing graphical representations, such as nuSketch [30] for making strategic decisions. In nuSketch, layers are used distinct to between the different categories of domain information.

2.4 Graphics

External representations can have many forms. Graphics² are a subset of external representations. External representations are not graphics by default, for instance, physical objects are also external representations [7]. The topic of graphics is diverse and complex. It covers many research areas that varies from medical applications to generic database visualisation.

The subject of graphics is discussed by the following themes:

- How do we apply graphics in education?
- How to deal with graphical complexity?
- How to build an effective graphical user interface?

2.4.1 Application of Graphics in Education

Rieber [51] discusses how graphics can play a role in instruction. Basically, Rieber distinguishes three categories of graphics. First, *representational graphics* are graphics that show a resemblance with the objects they represent. Second, *analogical graphics* refer to the known to explain the unknown. Analogy can be used as a building block to construct a new concept. Finally, *arbitrary graphics* are all graphics that do not fall into the other two categories. These graphics have visual and spatial characteristics that convey meaning.

The instructional applications of graphics can be described using two functions:

- Affective functions:
 - Cosmetic: graphics are used purely for decoration purposes, no direct learning benefits can be expected from cosmetic graphics.
 - Motivation: graphics used to raise the general motivation level, and is often due to novelty. This *novelty effect* diminishes over time, and so do the motivational effects.
- Cognitive functions:
 - Attention gaining: graphics are a source of stimuli that compete with many other stimuli. Graphics are often used to gain and keep the learners attention to instructional tool. Attention gaining graphics are an obvious, practical and rational use of graphics.
 - Presentation: graphics can be used to accompany text. It can be used to demonstrate or elaborate a concept.
 - Practice: real time animated graphics and simulations can be applied in an interactive learning environment [20] to provide means of training and practice. Flight simulators are examples of this category.

2.4.2 Dealing with Complexity

The problem of dealing with complexity is common in all scientific areas concerned with representations. As discussed below, different viewpoints supply different solutions.

²Graphics and visualisations are common words for almost the same thing. Graphics depict the phenomenon of images, while visualisation depicts the process of making them. In this text we will use the word graphics and make no distinction between the two concepts.

Visual languages

Woodruff [63], is concerned with visual languages and computing. He describes the concept of *semantic zoom*. Multiple related database tables are displayed by (nested) visualisations. Semantic zoom systems typically use a metaphor in which objects are displayed in a two dimensional canvas, which is viewed through a camera by the user. The user can pan and zoom this camera to view the different parts of the canvas. The height of the camera above the canvas is known as the *elevation*. Semantic zoom differs from normal graphical zoom in the sense that different graphical representations are used at different levels of elevation. In semantic zoom, certain views are deployed for different levels of detail. Certain details can be hidden and others made salient.

Information visualisation and navigation

Herman and Marshall [36] discuss the problem of complexity from the viewpoint of graph visualisation. They discuss various layout methods for graphs and argue that navigation and interaction facilities are essential in information visualisation. Several methods are discussed. As mentioned above, the zoom and pan functionality has two kinds: graphical and semantic. Both have the disadvantage of just displaying one scale. Suppose one has a map that can zoom and the city of Amsterdam is viewed in detail, it is impossible to get a detailed view of Milan without zooming out first and then zooming in again. To zoom without losing all contextual information, *focus+context* [32], techniques are used. Graphical fisheye views are popular techniques for displaying information without losing context. Another way of dealing with complexity is *incremental exploration* [13], techniques. This means that the system places a visible "window" on the graph, similar to what pan does, exploration means to move this window, also called *logical frames* [38], along a trajectory. Each direction the user decides to explore, a new frame is generated. In the most simple form, the same layout is used for every frame. More elaborate strategies are to enable the user to control the parameters that direct the layout algorithms. All these methods enable the user to zoom and navigate in different fashions, and tackle the symptoms of complexity, but they do not tackle the cause: the fact that there is too much information displayed to comprehend.

Strategies for reducing the number of visible items on the screen attempt to tackle the cause of the problem, that is, too many objects to visualise. Various strategies of "abstraction" and "reduction" have been applied by researchers in order to reduce the visual complexity of the graph. One approach is *clustering*. Clustering is identifying groupings or classes of data based on chosen semantics. Two different types exist, *content-based clustering* and *structure-based clustering* [52]. Content based clustering is highly specialised to the problem, whilst structure based clustering is more generalisable. In clustering certain nodes get emphasised while other nodes get suppressed. The latter can be done by *ghosting*, *hiding* and *grouping* [41]:

- *Ghosting* de-emphasising nodes, or relegating nodes to the background.
- *Hiding* simply not displaying the nodes.
- *Grouping* grouping nodes under new super-node representation.

2.5 Support of External Representations

External representations are usually employed in integrated environments. Their success depends on the integration with their environment. First of all, the entire user interface in general must facilitate the knowledge articulation process. This is the first topic that is addressed. Besides the user interface that must obey up to certain requirements, there must be support. This can range from providing help text to intelligent agents that guide the user through the construction process. This is the second topic that is addressed.

2.5.1 Graphical User Interfaces

Preece [49] discusses three aspects that describe the operational nature of a graphical user interface. First, the state of the system must be visible. Second, the system needs to make clear what kinds of interactions are possible. Finally, the system's feedback is important.

Nielsen [45] concentrates on how to evaluate software. Besides a cognitive walk-through, Nielsen proposes a heuristic approach, high level guidelines on how to evaluate a system. These are:

Visibility of system status The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

Recognition rather than recall Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognise, diagnose, and recover from errors Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Although these heuristics are used for evaluation, they should be kept in mind when designing a graphical user interface.

2.5.2 Support and Agents

Besides providing modelling environments itself, providing support is of key importance to support model building. This section discusses how providing intelligent support can improve the users performance and satisfaction.

Types of support

Preece et al. [49] argue that it is hard and difficult to get empirical evidence for the effectiveness of support and help systems. The added value of help systems is hard to measure. To provide adequate help, it is necessary to realize *why* and *when* users need support. Preece et al. enumerate the questions in users' mind that prompt the use of help systems:

- goal exploration: what can I do with this program?
- definition and description: what is this and what is it for?
- task achievement: how do I do this?
- diagnostic: how did that happen?
- state definition: where am I?

Note that the knowledge needed for providing support to the first three questions have a static nature. The goal of the program, the definitions and descriptions, and the task achievement can be described without any underlying knowledge model. Providing support for these kinds of questions is quite straightforward and can be done by providing the right texts. On the other hand, diagnosis of a problem and state definition need a underlying model representation in order to provide support to the user. This kind of support is more complex and needs reasoning to provide support.

How to interact

Fischer [26] argues that uses of computers in education has focused on two major approaches: *intelligent tutoring systems* and *interactive learning environments*. The strength of intelligent tutoring systems lies in the ability to control the curriculum by combining domain knowledge with a *student model*. This kind of tutoring is successful with learners that are new to a certain domain, but of little help for intermediate learners. Interactive learning environments such as proposed by Papert [47] do not have the problem being too strict in their guidance. But they provide limited support in helping learners to detect mistakes and overcome breakdowns. Fischer proposes the idea of *domain-oriented design environments*. A domain oriented design environment is a collection of interrelated tools and information repositories that provide specific support for communicating about and exploring concepts within the domain. First, these environments tackle the limitations of intelligent tutoring systems and interactive learning environments. Second, these environments provide multiple learning opportunities. Fischer describes five components of domain-oriented design environments:

- The *construction component* is the principal medium for modelling and design. It provides a palette of domain-oriented design units. These design units represent the primitive elements in the construction of the design.
- The *specification component* allows designers to describe abstract characteristics of the design they have in mind.
- The *argumentative hypermedia components* provides the users with a media to exchange knowledge with others and argument about their models.
- The *catalog component* provides a collection of previous constructed designs. These are examples and the components support reuse of components.
- *Simulation mechanisms* support users in understanding of the behaviour of the system.

The ideas of Fischer are important because he stresses that a building environment can consist of many tools and aids that are complementary. Furthermore, he points out that domain-oriented support tools are necessary to guide people in their modelling progress.

Agents for providing support

A complete other angle on the subject of supporting users is the agent approach. Bobrow [8] argues that agents are not just simple deaf and blind agents, helped by humans who formulate their problems in a predefined language. Agents participate in their environment by interacting and communicating with other agents. He describes three dimensions of interaction:

Communication If two agents communicate, there must be a kind of *common ground* of mutual understanding. An agent can help creating and maintaining this common ground of understanding. Furthermore, this basis for communication can be difficult. To facilitate this communication, an agents can be used to *mediate* the communication.

Coordination With multiple agents with multiple goals, progress requires to share resources and have a common goal. Various organisational structures require resources to be shared. But, furthermore, *joint commitment* to future action is a useful way of letting independent agents work together.

Integration For artificial intelligence (AI) systems to provide solutions in the real world, they need to be integrated in human work practices. Systems must solve problems that have a payoff outside the system itself. For agents to be useful, they must fit in with our work practice of humans and existing computer systems. The use of *expert advisers* integrated in traditional programs will improve people's skills and adaptivity in their work practice.

According to Bobrow agents can support our model building tasks in different ways. Agents can communicate knowledge by acting as a mediator. Agents can act as expert advisers to provide us with diagnosis and problem state descriptions.

We now can combine the advocated ideas on how to aid model builders. Preece defines what kinds of support are necessary. Fischer argues that different tools and sources of information can aid the model builder in the modelling task. Bobrow visions agents helping humans in communicating. Agents integrate systems with humans by acting as expert advisers.

2.6 Qualitative Simulation and Modeling

In the previous section external representations have been discussed. External representations are deployed to aid model builders in articulating their knowledge. In this section detailed aspects of the modelling language are discussed. Qualitative reasoning is a language or formalism in which knowledge can be expressed [14]. This knowledge is articulated in models which behaviour can be inspected and analysed by running simulations. First, the reasons for using qualitative modelling in educational contexts are discussed. Second, qualitative reasoning is explained in detail. Finally, the need for external representations in model building and simulation is addressed.

2.6.1 Simulations vs. Model Building

Computer based simulation is becoming increasingly common. In a simulation the learner is presented a model, and invited to explore it. The model is build by an expert. The user has the freedom to adjust certain parameters and explore the consequences of the changes. The ability to run simulations or pre-built models is a vast improvement over text books, but the principle of learning by construction is ignored. To make powerful use of models Wilensky [62] argues, learners must first build their own models and develop their own investigations. To support users in building (qualitative) models, a number of modelling environments have been put forward.

2.6.2 Why Qualitative Modeling?

Forbus [27] advocates modelling in the context of qualitative reasoning. Modelling is a central skill in scientific reasoning. He puts the role of modelling in education as:

"Learning to formulate, analyse, test, and revise models is a crucial aspect of understanding science, and critical in helping learners in becoming active, lifelong learners".

Forbus distinguishes three important reasons why we want learners to become modelers. First, models provide a mean to externalise thought. Second, the process of modelling itself can be of value for the learners. And finally, it provides practice with formal representations. Below these three reasons are further explained:

Externalising thought Providing the learner with external representations help reduce the working memory load. Learners are able to comprehend more complex problems than they could otherwise. Externalising thoughts also provides means for communicating the formalised knowledge to facilitate collaboration and discussion [50]. Peer to peer questioning, discussion, and justification has proved to be productive in aiding learning among young learners [16]. Expressing oneself in a formal language helps in being able to understand more and more complex problems, and facilitates communicating them with others.

The process of modelling By forcing learners to articulate their beliefs into a structural framework, they will develop a better understanding of the phenomena being modelled. Furthermore, they will develop a broader understanding of complex, interrelated systems. Expressing a system in qualitative reasoning principals will force learners to think in terms of causal relationships. The notion of causality in modelling should be part of the standard curriculum.

Practice in formal representations By building models, learners will get practice in using a formal representation, a skill needed for mastering programming and mathematics. This will provide them with more experience and practice in technical vocabularies. Qualitative modelling can provide a bridge to more complex and symbolic formalisms. Learners will become acquainted with the process of creating, testing, debugging and fixing. This kind of experience may stimulate a better understanding of many computer related formalisms.

Jonassen [40] argues that experienced problem solvers possess tightly related qualitative and quantitative representations. Expert solving physics problems perform a qualitative analysis of the problem before attempting a quantitative analysis. Qualitative representations support quantitative solutions problems. It thus is important to teach students how to represent problems qualitatively, because qualitatively represented problems improve students performance.

Bouwer et al. [10] argue that quantitative, mathematical models have proven their usefulness in education, but mathematical models are not always sufficient. First, precise information can be missing and, second, it needs considerable knowledge of mathematics. Even more interesting than overcoming the limitations of mathematical models, are aspects of qualitative models have for specific learning activities. As mentioned by

The fact that qualitative models capture causality has pleasant side effects. The model itself, and the behaviour of the system can be reasoned about. The qualitative model can act as a domain model in intelligent learning environments. Currently, there is work done by Bouwer [9] on explanation of qualitative simulations. The system can not only display its behaviour, but also explain to the user why this behaviour occurs. Furthermore, the computability of the formalisms in qualitative modelling allow for intelligent tools like automated question generation [34].

2.6.3 Qualitative Reasoning

Qualitative reasoning is discussed by its four key concepts: causality, quantitateness, reasoning from structure, and the state graph. The first key aspect of qualitative reasoning is *causality*. Causality allows modelers to express the causal relations representing the behaviour of the systems they model. Causality is an important concept in human reasoning. Being able to convey causality enables qualitative reasoning to model common sense.

The second key aspect of qualitative reasoning is *quantitateness*. In mathematical models, each variable, or quantity, needs to have a numerical value. Qualitative reasoning does not use numbers, instead,

it uses points and intervals to depict a value of a quantity. As a result of this, qualitative models have a level of abstraction. The modeler is forced to describe the quantities in *qualitative* terms instead of quantitative numbers. This frees the modeler from being forced to assign numbers to values. Quantitativeness is essential concept in qualitative reasoning.

The third key aspect of qualitative reasoning is *reasoning from structure*. Behaviour of the system is associated with the structural features of the system. Behaviour of the system is modelled in rules, or model fragments. These model fragments always need a structural feature to apply for. It is not possible to solely refer to behaviour. Modelers are forced to perceive the structural features of the system as basis for all behaviour prediction.

The final key aspect of qualitative reasoning is the *state graph*. The state graph specifies the set of possible behaviours. Instead of one single solution to the prediction problem, the system displays all solutions it finds; the system is not deterministic in its prediction. This means that in case of unspecified behaviour, the system will reason using all possible values. A simulation seldom consists of one single answer to the prediction problem, but consists of a set of different solutions.

2.6.4 Qualitative Prediction of Behaviour

Qualitative prediction of behaviour consists of three global tasks [14]: modelling, prediction and interpretation. First, a model is created by a modeler. Second, an inference engine will calculate a prediction of the model by identifying distinct states of behaviour that the system can reach in the course of time. And third, the result of the simulation must be interpreted by a kind of agent to analyse the output and classify the behaviour classes of the output.

Figure 2.2 depicts the prediction process. Note that it takes as an input the scenario, the library of model

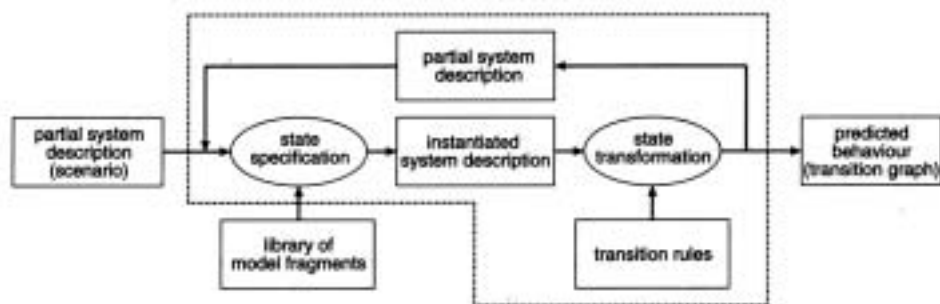


Figure 2.2

Qualitative Prediction of Behaviour taken from [23].

fragments and the transition rules.

Bredeweg [14] describes the qualitative prediction of behaviour. The scenario acts as the initial situation for the simulation. The model fragments that apply to the initial situation are combined into a *instantiated system description*. The termination rules which govern the possible changes will be applied and new successive states may be generated into a *partial system description*. These successor states are again input to the specification process. This process ends once no more new states are generated. The final output of this cycle consists of the *state transition graph* comprising all different qualitative states the system can be in, as well as the valid transitions between these states.

Model Fragments

Model fragments consist of conditions and consequences:

Conditions The conditions are the requirements that have to be met in order to make a model fragment applicable. This resembles the *if* part in an *if-then* rule. Conditional to a model fragment can be: entities, relations, quantities, values, inequalities and other model fragments.

Consequences The knowledge that is applicable once the model fragment's condition holds. This can be viewed as the *then* part of the *if-then* rule. The consequence of a model fragment can contain entities, relations, quantities, values, inequalities and causal dependencies.

Conditions and consequences thus consist of structural parts and behavioural parts.

- Structural primitives

Generic entities The elements that make up the physical system are described as instances of generic entities. The generic description of entities are classified by means of an isa hierarchy.

Structural Relations The structural connections between the various entities. They also can be instantiated to make up a physical system.

The design principle of *no function in structure* [22] [21] argues that no functional knowledge may be part of the structural part of the model.

- Behavioural primitives

Quantities The behavioural features of entities, also referred to as variables or parameters. A quantity in a simulation has a value and a derivative.

Quantity spaces Each quantity has a specific value range. A sequence of intervals and points. This is an abstraction of the "real" value range. Note that this is a key concept of qualitative reasoning in the sense that this is the qualitative description of the system. No real numbers are used, but instead, an ordered set of labels depict the possible values quantities can take. For instance, if one wants to model the behaviour of water. Figure 2.3 displays a possible

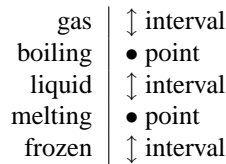


Figure 2.3

A quantity space of water..

interpretation of how water "behaves". This quantity space has two points: melting and boiling. Furthermore, there are three intervals: frozen, liquid and gas.

Derivative Besides a value, a quantity has a derivative. The derivative can be increasing, stable and decreasing. The derivative depicts the direction of change for the current quantity.

Dependencies The behavioural relations between the different quantities. These are the behavioural relations that influence the quantities and change their values. Dependencies can be divided into *causal dependencies* and non-causal dependencies *Inequalities*

Inequalities Inequalities represent constraints on or between quantities. They are: equal (=), greater (>), smaller (<), greater-equal (\geq) and smaller-equal (\leq). Besides for quantities, inequalities may be specified for values and derivatives. Inequalities influence the behaviour of the system by constraints. For instance, if an inequality states that two values of two quantities must be equal, all states must comply to this constraint.

Causal dependencies Causal dependencies depict causality, as their name suggests. They can be of different types:

- *Correspondences* express that two quantities Q_1 and Q_2 have corresponding values: for every value of Q_1 there is a corresponding value of Q_2 . Correspondences can be directed or undirected. In *undirected correspondences* the correspondence works both ways, from Q_1 to Q_2 and also for Q_2 to Q_1 . For *directed correspondences* it only works from one to another, thus from Q_1 to Q_2 and not visa versa. Furthermore, there can be correspondences between values and quantity spaces. Note that the quantity spaces of the two quantities must be similar in order to have a correspondence.
- *Proportionalities* denotes that the direction of change of Q_1 influences the direction of Q_2 . Thus, if Q_1 increases, Q_2 does to.
- *Influences* are used to express that the value of a quantity determines the derivative of another quantity. If Q_1 influences Q_2 , Q_1 is greater than zero, Q_2 is increasing. Derivatives can be *positive* or *negative*. Note that the quantity space of Q_1 needs to have a value named zero in order to have an influence on Q_2 . It must be possible to deduce if a value is positive or negative. To do so, there must be a value acting as "zero" in the quantity space.

Transition Rules

The relation between states is modelled by termination rules. Three types are distinguished:

- *Termination Rules* specify how qualitative states may end. This may be by a quantity changing value or by a changing inequality between quantities.
- *Ordering Rules* specify in what order the changes defined by the termination rules will take place. Ordering rules are also used to merge changes that constitute one transition.
- *Continuity Rules* deal with the quantities that do not change value during a state transition.

Scenario

In order to simulate the behaviour of a specific system, there needs to be an initial description of the system. This is the input system or scenario. A scenario contains entities, quantities, quantity spaces, values, derivatives, and inequalities.

State Transition Graph

The state transition graph is the result of the simulation. It consists of:

- States consisting of the quantities, values, and causal model of the simulation in that specific state. The causal model consists of the dependencies that are active in the current state.
- Transitions are the transformations between the states. They consist of the transition rules that fired in order to generate the succession state.

2.6.5 Visualisation in Qualitative Reasoning

As discussed by Bredeweg [14], qualitative reasoning consists of three global tasks: creating the model, predicting the systems behaviour by simulating the system, and inspecting the results of the simulation. The first and the latter tasks involve human agents, simulating the system is automated.

Graphical representations can be used to aid the modeler in building its model and inspecting the result of a simulation. The distinct structure of the qualitative reasoning primitives can be exploited to create useful external representations that aid the modeler in the modelling task and the inspection task. For instance, the hierarchical aspects of an entity hierarchy elicits a connected graph representation. Besides aiding model building by providing representations that map directly on the qualitative reasoning primitives, representations can deliver a number of abstractions. Bouwer et al. [10] mention the idea of *horizontal views*, views

that provide a global overview, and the idea of *intermediate modelling support*, support for the articulation of intermediate models in another formalism than the qualitative reasoning primitives. Furthermore, as discussed in section 2.5, different types of agents providing support and communication can aid the model builder in the modelling task. Bouwer et al. [10] discuss the idea of *model building support from a 'content' point of view*, support should go further than just checking the syntax, it should give intelligent reflections on the model building task.

High Level Tools for Building Models

3.1 Introduction

A model building effort consists of model creation, behaviour prediction, and simulation inspection. This chapter concerns the first task: model creation.

First, the existing model building environment and the architecture of the system is discussed. The model building environment consists of *construction components*. Construction components are tools that provide the modeler with the basic model building constructs for building qualitative models.

Second, this chapter covers two *specification components*. Specification components are tools that support the model building process by allowing for articulation in a different, often more unrestricted, formalism than the qualitative language formalisms. The first specification component is a tool that facilitates sketching. The second component is a tool that facilitates in causal model creation and testing.

3.2 General Architecture

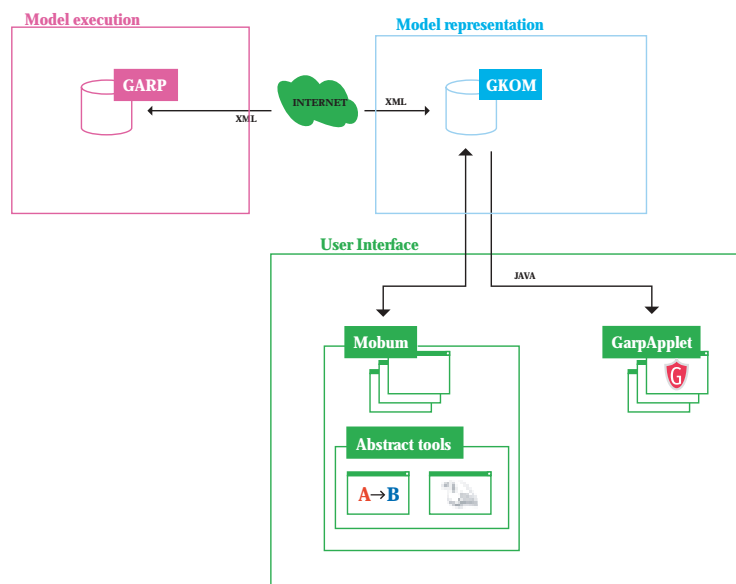


Figure 3.1
The architecture of the entire framework.

MOBUM [5] is part of a larger framework, displayed in figure 3.1. This framework consists of:

Name	Description	Section/Chapter
GARP Server	The server that executes the models by running simulations. The server has a library of existing models.	section 3.2.2
XML specification	The definition of a XML specification used for communicating the models, simulation control, and the results of the simulation.	section 3.2.1
GKOM Library	The communication library that uses XML to communicate with the server and holds a symbolic representation of the knowledge in a GARP model and simulation.	section 3.2.3
MOBUM	MOBUM is a set of construction components that map on the GARP ontology.	section 3.3
specification tools	Causal Model Pad and SWAN are two specification components.	section 3.4 & 3.5
GarpApplet	The simulation inspection tool.	chapter 4

3.2.1 The XML Specification

The XML specification represents the GARP model in XML. As depicted in figure 3.1, the GARP server communicates using a XML specification. XML is used for several reasons. First, the ability to represent hierarchical data structures makes it easy to articulate GARP knowledge in XML. Second, XML parsers are available in almost any programming language and on almost any platform. Third, XML can make use of DTD's, which provides a structure definition, and subsequently, syntax checking. Fourth, XML is purely textual and easy to transport through common transportation protocols. The XML communication in this framework is based on the HTTP/1.0 ¹ specification, allowing for a great level of flexibility².

3.2.2 The GARP Server

The GARP server is a wrapped GARP simulator written in Prolog. It is controlled from a HTTP server and listens to a specific port. The GARP server parses the incoming HTTP request from the client into a command for the simulator. The simulator executes the command and infers the result. The result, often a simulation, is translated into XML and included in the HTTP response to a client's request.

3.2.3 GKOM

GARP Knowledge Object Model, or GKOM [59], represents a GARP model in JavaTM. This library, or application program interface (API), is not a stand alone application, but a library that proxies all important aspects of the GARP server. The GKOM library is able to parse and generate this XML into its own representation. This representation is, in contrary to GARP itself, an object oriented representation. This facilitates the accessibility of the GARP server in an object-oriented language such as JavaTM. MOBUM, GarpApplet and the Causal Model Pad use GKOM as their server representation.

3.2.4 Client/Server Model

The GARP server, the GKOM library, and the XML specification make communication through the internet possible. This type of communication model is known as a client/server model. The client starts an

¹available at <http://www.ietf.org/rfc/rfc1945.txt>

²For instance, a simulation can be run using a web browser

interaction, or session with the server, in order to request a service from the server. This client/server model generally has two advantages:

- *Centralised computation.* The computational load³ is handled by the server, and does not stress the client computer. Since the computational software runs in a centralised fashion, it is easy to maintain and update without updating the client software.
- *Knowledge sharing* The current framework currently supports a centralised model base. However, the architecture facilitates elaborate knowledge sharing by allowing for exchanging model parts.

Major drawback of the client/server model is the network dependency. The architecture assumes a permanent on-line client, connected with a reasonable bandwidth⁴ to the internet. The server acts as a single point of failure, one server failing can obfuscate many clients. The server architecture is currently not equipped with any failover mechanisms, except for the client allowing for selecting different servers. The simulation's session-state maintenance is realised by a TCP/IP connection. Certain routers tend to drop this connection when unused for a certain amount time. This problem is tackled by a keep-alive thread which sends a keep-alive message⁵ on a fixed interval. Since the keep alive thread generates overhead, it is optional.

3.3 MOBUM

MOBUM [5], is the descendant of HOMER, and strongly influenced the evaluation of HOMER [39]. The user evaluation of HOMER highlighted a number of inefficiencies concerning the user interaction. Furthermore, user support and portability of the application became an issue. These newly gained insights are taken as guidelines for the design of MOBUM.

Major differences between HOMER and MOBUM

The full detailed discussion of, and rationale governing, the design of MOBUM is covered by Bessa-Machado [6]. The major differences are:

- *Improved user interaction.* HOMER confuses the users because it has inconsistencies and a non-intuitive interaction. MOBUM attempts to offer a more consistent user interface and a more visible system status. For instance, HOMER allows adding items in the model fragment only by menu. MOBUM offers three ways throughout the entire user interface: the drop down menu, buttons on a toolbar, and context menu's. Furthermore, MOBUM disables options if they do not apply.
- *The ability to run the simulation* and inspect the result. The integration with GarpApplet enables the user to directly run the created model. In the case of HOMER, the user needs to save the model to disk, which, in turn can be run and inspected using a separate program, namely VisiGarp [11]. The GarpApplet is discussed in chapter 4
- *The user support.* Help files and intelligent agents support the model builder at different levels. The support and intelligent help agents are discussed in chapter 5.
- *Abstract tools.* As discussed in this chapter, different specification components support the model builder in conceiving vague, early, or intermediate ideas into a specification.

Figure 3.2 displays the model fragment construction tool in HOMER. Figure 3.3 displays MOBUM's equivalent. One distinction between the interface of HOMER and MOBUM is the toolbar on the left side of the builders in MOBUM. MOBUM uses toolbars to present the actions available to the user. Another obvious distinction between the two interfaces are the avatar icons in the right of the builders in MOBUM. They

³Simulations can create a very heavy computational load, up to 30 seconds on modern machines.

⁴Large simulations can take up to 2MB of communication, taking more than 2 minutes on a 14k4 modem

⁵Sometimes referred to as a heartbeat.

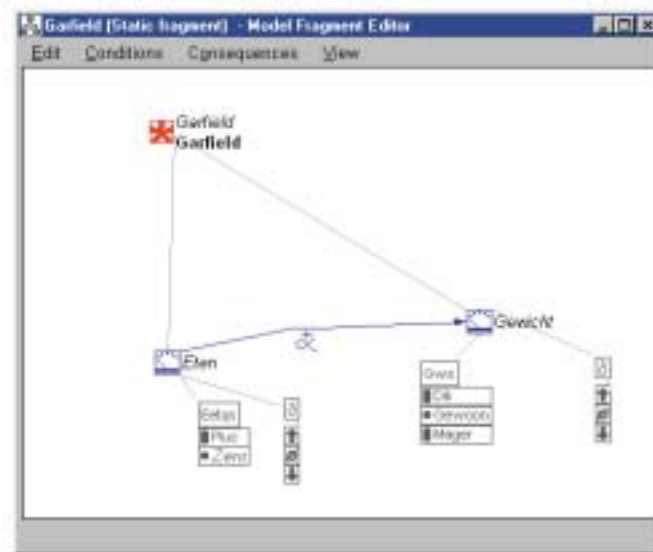


Figure 3.2

HOMER displaying a Model Fragment from the Garfield assignment.

give explanation and reflect on the current model. These are discussed in more detail in chapter 5. Thus, in MOBUM things can be added by drop-down menu, toolbar and context menu. This is done consistently throughout the application. Furthermore, toolbar buttons that cannot be applied are disabled. This is visualised by graying them out. This supports the visibility of the system state, which is an important aspect of human-computer interaction, as discussed by Preece [49]. The reaction of the user interface to the user's selection provides information for the user about the actions possible to the current selection. For instance, the dependency button remains disabled until a selection of two quantities is made. Drawback of this approach is that the user must be aware it can make selections and that selection can consist of multiple items.

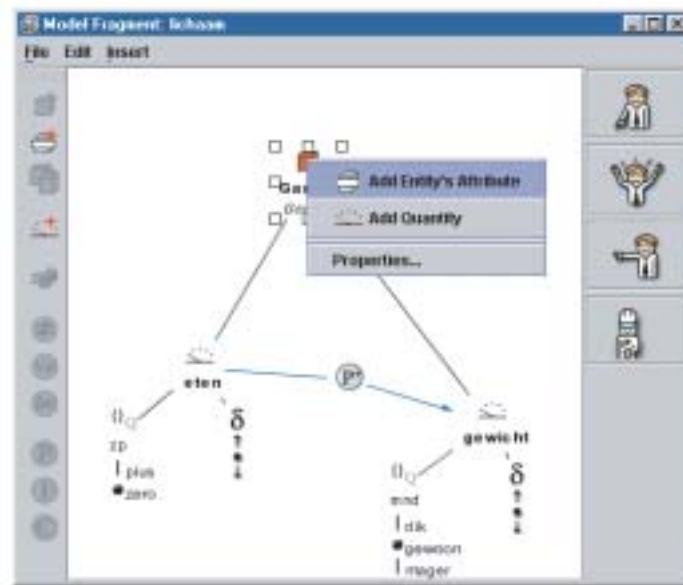
3.3.1 The Tools

MOBUM consists of a collection of construction components, mapped closely on to the ontology of qualitative simulation. These tools are:

- *Structure Builder* for modelling generic entities and relations.
- *Quantity Builder* for creating quantities and linking them to quantity spaces.
- *Quantity Space Builder* for creating quantity spaces.
- *Model Fragment Builder* for creating the model fragments by using instances of the entities, relations, quantities and quantity spaces and describing behaviour by inequalities and causal dependencies.
- *Scenario Builder* for creating the input systems.

3.4 The SWAN Sketchpad

The SWAN (Sketchpad Without A Name) sketchpad provides a sketching environment. The SWAN sketchpad is a drawing tool, extended with tools to name and type objects in the drawing. Furthermore, the SWAN sketchpad allows for behaviour prediction.

**Figure 3.3**

MOBUM displaying a model fragment from the Garfield example.

3.4.1 Rationale

The SWAN sketchpad provides the modeler with a drawing tool for sketching purposes. It enables modelers to make a quick and unconstrained drawing of their ideas. Sketching is viewed as an important (first) step in the model building process [58]. Sketches play an important role in many design processes, this is discussed in section 2.3.4. A sketch is abstract and inexact. The rationale behind the sketchpad is that modelers must be facilitated in expressing their ideas easily. Providing a unconstrained sketching environment facilitates this. Subsequently, modelers can become more formal in their expressions. The SWAN sketchpad supports this process of becoming more formal. First, it enables for naming. The graphical objects drawn can be named by text labels. The SWAN sketchpad can, based on the position of the text label, infer which object the text label belongs to. Second it allows for typing. Graphical objects in the drawings can be typed. The type of a graphical object is displayed by a small icon.

The SWAN sketchpad is integrated in the modelling environment. Agents, discussed in section 5.6, aid the modeler in the model building process. One way of supporting the modeler is to make the information in the SWAN sketchpad available in the appropriate construction components in the modelling environment. Information expressed in the SWAN sketchpad does not have to be expressed again when actually building the model.

Furthermore, the created drawing can be used to envision the behaviour of the model. A filmstrip metaphor is employed to enable the modeler to create multiple 'scenes' to graphically envision how the behaviour of the system will be. A 'scene' is created by cloning the original drawing. The modeler can change, add, and remove objects in the newly created drawing. Each scene represents an important state the modelers expect the behaviour of the system to display.

3.4.2 Drawing, Naming and Typing

In the sketching phase of the model building process, the user is not restricted to any graphical notation. The first possible steps in becoming more specific are naming and typing. The process of drawing and specifying consists of three steps:

1. Drawing objects.

2. Assign a name to individual graphical objects using text labels.
3. Assign a type to individual graphical objects in the drawing.

These steps do not have to be performed in any order. The user is not obliged to do any of them in the model building process. The tool is meant for sketching and thereby delivers a number of inexact and abstract ways of expressing ideas about a model. But the tool does not represent a *necessary* step in the model building process. Nor does any of the specification steps.

A name can be assigned to a graphical object by attaching a text label to the object. This name can be used to refer to the object. All modelling primitives need at least a name. By assigning a name to a graphical object, the object can be imported by the construction components. If an object is not provided with a name, it will be referred to as "unknown". A type is assigned to a graphical object in order to inform the model building environment about the type of context the object should be applied to. For instance, a graphical object typed as "object" will be mentioned in the Structure Builder, since entities are the modelling primitives that "objects" refer to. A graphical object typed "quantity" will be mentioned in the Quantity Builder, since quantity is the modelling primitive the type "quantity refers to".

By specifying name and type of the objects in the drawing, the modeler becomes more formal. The actual process of transferring this knowledge to other builders is discussed in chapter 5, here is described how a rule engine infers the knowledge from the drawing and how a framework of agents transports this knowledge through the modelling environment. The upshot of this tool is that the user is provided with a free drawing environment, and a set of tools to undertake the first steps in becoming more formal. The knowledge expressed in the SWAN sketchpad can be transferred to the appropriate construction tool in MOBUM.

Drawing Tools

The user has the availability over a set of drawing tools to make shapes. These tools are:

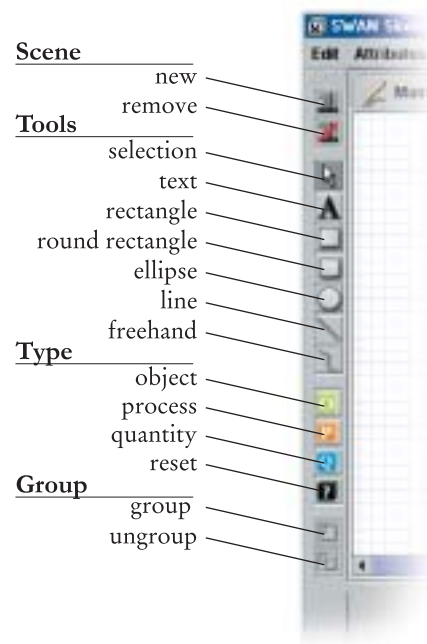
- *rectangle tool* for drawing rectangles.
- *round rectangle tool* for drawing rounded rectangles.
- *ellipse tool* for drawing ellipse shaped objects and circles.
- *line tool* for drawing straight lines.
- *free draw tool* for creating arbitrary lines.

Each graphical object has a line colour and a fill colour. A line can have a width, which is null if an object does not have a line. An object can have a texture, proving the object fills an area. For instance, a line cannot have texture, since it does not fill an area. The drawing environment is equipped with a grouping tool, that groups different into one group that can be treated as one object. Grouping can be undone by the un-group tool. Figure 3.4 displays the toolbar. Figure 3.5 displays a U-Tube drawn in the SWAN Sketchpad.

Naming Tools

Naming can be done by the text-label tool. Existing text can be edited. The font and style can be changed and the colour of the text can be modified. Existing text can be edited. Assigning a name to a graphical object will be used to refer to the object when distributed through the model building environment. In order to annotate a graphical object, a text label must be connected somehow to the graphical object. This connection is inferred by the SWAN sketchpad. By inferring this knowledge, the user is able to assign names in a flexible fashion. This inference is governed by three conditions. These three conditions on which a text label can name an object are:

1. The text label must overlap the object. Putting a label on top of a graphical object will assign the name to this object.

**Figure 3.4**

The swan toolbar and the topology of the different icons..

2. The text label must be connected by a line with the target object. Drawing a line between the text label and the target object will assign the name to that object.
3. The text label must be grouped with the target object. This is a more elaborate function. The grouping functionality enables the modeler to enforce the assignment of the text label to a certain graphical object.

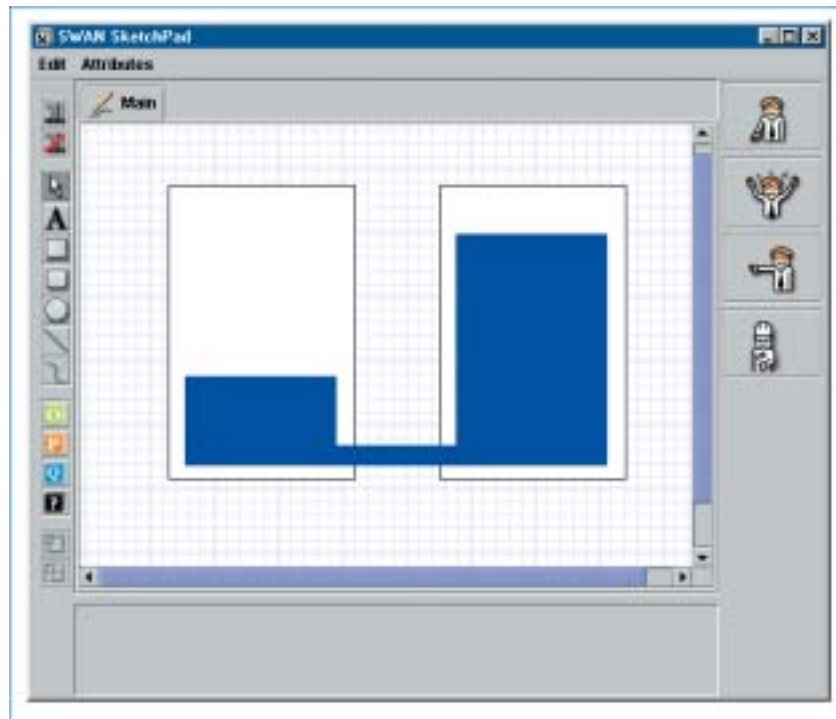
If one of these three conditions is met, the name of the label will be assigned to the target object. Figure 3.6 displays the three conditions in which the name is connected to the target object. "Water" uses the first, that is, it overlaps its target. "Flow" uses the second condition. The line connects the label and the target. The "Container" label and its target object is grouped. Groupings are not visible unless a selection is made, figure 3.6 displays the grouped object selected.

Typing Tools

tools can be typed in three classes:

- Objects
- Processes
- Quantities

These classes each have their own icon, that will be on top of the typed object. Figure 3.4 displays the typing tools. The fourth, black, tool is for removal of the typing. Figure 3.7 also displays the typed U-Tube. The container is typed as an object. The flow is typed as a process and the water is typed as a quantity. Note that the drawing does not and should not depict which entity the quantity belongs to because that kind of specifics is not necessary at this stage of the model building process.

**Figure 3.5**

The SWAN interface with an initial drawing of the U-Tube.

3.4.3 Behaviour Envisioning

Besides creating a sketch drawing and specifying the basic types and names of the objects in the model, the SWAN Sketchpad facilitates in envisioning behaviour. The general idea is that different scenes represent the envisioned states that the user distinguishes. This enables the modeler to express ideas about the behaviour of the system and the states the simulation is expected to generate.

In the sketchpad, the metaphor of movie scenes is employed. Each predicted state in the simulation is represented as a scene. The modeler can make a copy of the current drawing and modify aspects in order to express expected behaviour. This envisioning makes the modeler's conception on how the simulation should behave more explicit. The SWAN sketchpad has one *original drawing*, or initial drawing. This is the drawing created in the sketching process. The filmstrip adds *scene* drawings to the SWAN sketchpad. It is important to distinguish between the two types of drawings; the original drawing describes the static features of a system, the scene drawings describe the change.

As figure 3.4 shows, there are two buttons for manipulating scenes, one to add a new scene based on the current drawing and one to remove the current scene. Creating a new scene will make a copy of the current drawing. New objects drawn in the original drawing will be propagated in the derived scenes: if a new object is created in the original drawing it will appear in all scenes. If new objects are created in a scene, it will not be automatically appear in any other scene or drawing. The type of objects can always be changed in the original drawing. All derived objects in the scenes will also change to the new type. Type cannot be changed in the scene drawings unless they are not typed in the original or do not exist in the existing drawing.

Figure 3.8 displays a simple behaviour envisioning: a unbalanced U-Tube becomes balanced in the second state. Notice the filmstrip metaphor that is used in the bottom half of the screen. The original drawing is omitted in the filmstrip to prevent the user from having misconceptions about the type of drawing. Misconceptions in the type of drawings are furthermore prevented by the use of a different background colour in the scene drawings.

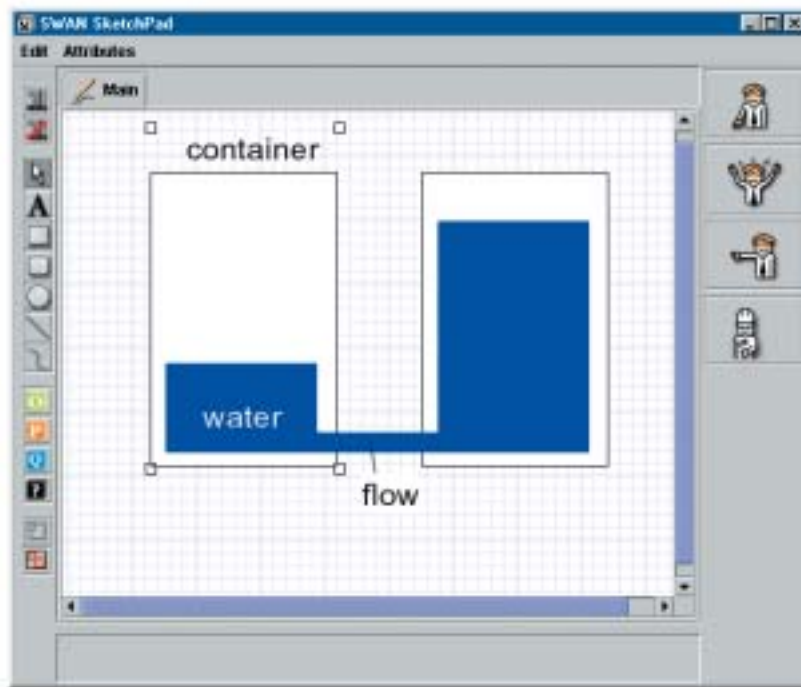


Figure 3.6

The SWAN sketchpad displaying the conditions on which the name is assigned to the object. "Water" is simply put on top of its target. "Flow" is connected with a line to its target. "Container" is grouped with its target.

3.5 The Causal Model Pad

The Causal Model Pad is a *specification component* for formalising and testing a causal model. The causal model pad provides the modeler with the following functionality:

- The Causal Model Pad provides the articulation of a causal model by using a restricted graphical syntax. Quantity spaces and entities, for instance, cannot be specified in the Causal Model Pad.
- The causal model pad is able to simulate the causal model it specifies. The causal model pad is able to generate a GARP model based on the specification, send the model to the server and display the result to the modeler.

These two aspects add unique possibilities to the modelling environment. First, the modeler is provided with an early and easy means of expressing a causal model. The causal model pad helps to formalise and specify the his causal model without being concerned with all other aspects of the model. The graphical constraints, as argued in [56], match those of the represented relations. Quantities can only be connected by dependencies, matching the structure of the causal model.

Second, the causal model pad provides feedback on the specified causal model. This short *feedback loop* enables the modeler to:

- reflect on the causal model in early stages of model refinement. This reflection enables the modeler to test and adjust the causal model in mind.
- the causal model pad can be viewed as a building environment that can be used in the process of learning by building as proposed by Papert [47].

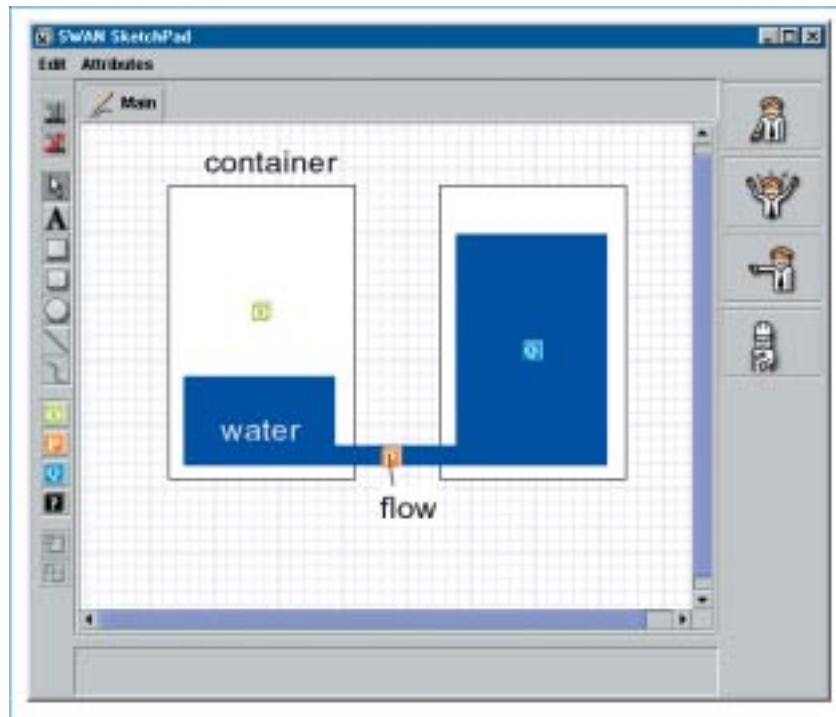


Figure 3.7

The U-Tube typed. The container is typed as an object, the flow is depicted as a process that takes place in the pipe, and the water in the container is described as a quantity..

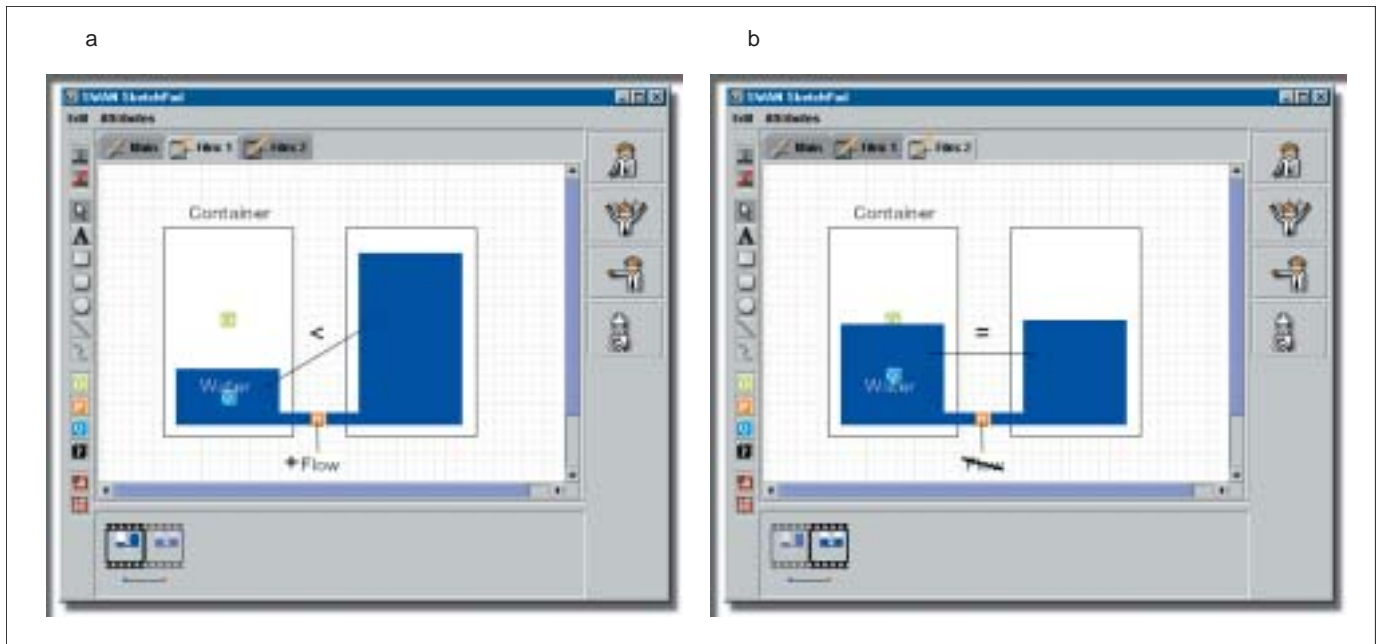
By using the causal model pad, the (unexperienced) modeler can learn the aspects of a causal models in qualitative reasoning. The causal model pad can be deployed as a stand alone model building environment for learners.

3.5.1 The Modelling Environment

The causal model pad supports the creation of quantities, inequalities and dependencies. Quantities are visualised as rounded rectangles, with their name on the left and their value space on the right. Figure 3.9 displays the causal model pad. Dependencies and inequalities are visualised as lines. An icon in the centre of the line depicts the type of the dependency.

3.5.2 The Simulation Inspection

Once the modeler has simulated the created model, the resulting states are displayed as different tabs. Figure 3.9b to d shows the result of simulation. The visualisation of the states do not differ from the model's visualisation, making it easier for the modelers to inspect the model. To prevent confusion, the names of the tabs differ, the background colour is different and, except for moving the quantities, editing is disabled. The bottom half of the screen displays the state navigation panel, which displays the simulation graph as a table. Selecting a state will display the corresponding state tab. The table view attempts to make the different transitions insightful. The transitions are visualised as lines starting at their from-state and ending at the to-state. In section 4.3.1 the table view is discussed in detail.

**Figure 3.8**

The behaviour envisioning of the U-Tube example. a) displays a unbalanced U-Tube with a flow, b) displays the end state, the levels are equal.

3.5.3 Model Generation

The Causal Model Pad only contain quantities, dependencies and inequalities. In order for the causal model pad to simulate the causal model, there need to be filled in the missing model parts in order to create an executable model. The causal model pad generates these missing parts into a *dummy model*. This dummy model is generated using the following assumptions:

- All quantities belong to one dummy entity.
- All quantities have the same quantity space: plus - zero - min.
- All quantities do not have a value unless it is explicitly set.
- All quantities, the dummy entity, the inequalities and dependencies are put into one model fragment. The entity and the quantities are conditional. The inequalities and dependencies are consequences.
- A scenario is created with the entity and the quantities with their optional values.

Once a model is generated, it is sent to the server for simulation. The result of the simulation is returned and visualised.

3.6 Evaluation

The SWAN Sketchpad and the causal model pad are based on new ideas about supporting the model building process. Apart from that, unconventional ideas about user interaction are deployed. In order to get feedback on these new tools an evaluation by expert users has been applied.

3.6.1 Research Question

The goal of this evaluation is to find out if the functionality that these tools offer is of use in the model building process. The question arises if these tools add useful additions to the model building environment. The mayor research question is:

Are the specification components useful tools in the model building process?

Besides this question there are a number of other issues:

- For what kind of users are the tools appropriate? Are these tools useful for experts or can they be useful for novices?
- What kind of problems can be expected and what kind of drawbacks does a tool like this have?
- What alternative uses are possible? Can these tools be applied in other areas than as a specification component in the modelling environment?

3.6.2 Review by Experts

Five modelling experts with prior experience with modelling software⁶ where demonstrated the tools and afterwards asked for their opinion. All experts where asked the same questions about each of the tools. They where able to respond freely and asked to make any comment they thought was appropriate. The evaluation of the tools is discussed by the functional aspects of the tools: sketching, behaviour envisioning, and causal modelling.

Sketching

The sketching and typing tools was generally seen as a useful tool which facilitates the model building process. The idea of being able to draw freely, while typing and naming functionality is at hand, appealed to all of the experts. The reusability of knowledge in the modelling environment appealed to most experts. The sketching functionality of the SWAN sketchpad is generally perceived as a bridge between concept and actual model. All experts thought the sketchpad can be a valuable tool in their own modelling efforts.

A number of interesting ideas have been mentioned:

- First, pre-created sketches can be used in educational contexts to accompany assignments. A picture can be used to guide the learner and explain ideas about the assignment.
- Second, sketches depict ideas about the model *at a certain point of time*. Sketches can act as documentation or as an archive of certain modelling ideas and believes. The trajectory of the design process can be archived into the different sketches such a process produces. The sketch archive can later on be used to track the progress of a modelling task. Decisions and their rationale can be preserved over time by sketches and their connotations. A similar idea is mentioned by Ullman [58].
- Third, the SWAN sketchpad can facilitate in the exchange of knowledge and ideas between modelers. As also mentioned by Ullman [58], sketches can be deployed to communicate ideas with other people.

Behaviour Envisioning

The evaluation of the behaviour envisioning, or filmstrip tool, was less unanimous as the sketching functionality of the SWAN sketchpad. A number of experts did not see very much use in the tools while others saw the behaviour envisioning as key in the scientific *hypothesise* \rightarrow *predict* \rightarrow *test* cycle.

A number of interesting ideas where mentioned.

- First, there should be support for defining the quantity spaces, or at least milestones. This is currently not facilitated.

⁶All experts are familiar with the HOMER environment.

- Second, there is currently not any inference possible on the drawing. If, for instance, the colour of an object is changed over different states, there is information that can be inferred from this. A modeler can specify a specific change in the system. This specific knowledge can be reused and made available in other parts of the modelling environment.

Mayor critique consisted of the applicability of this tool in the modelling process. A number of experts felt that predicting the behaviour is something that the simulator should do, and should not be done at the initial stage of model building. Although this is an interesting remark, it still remains interesting to let the modeler think in advance of the states that will be expected.

Causal Modeling

The causal model pad was highly appreciated. Almost all modelers believed they would use it in their model building efforts. The evaluation of the tool is twofold.

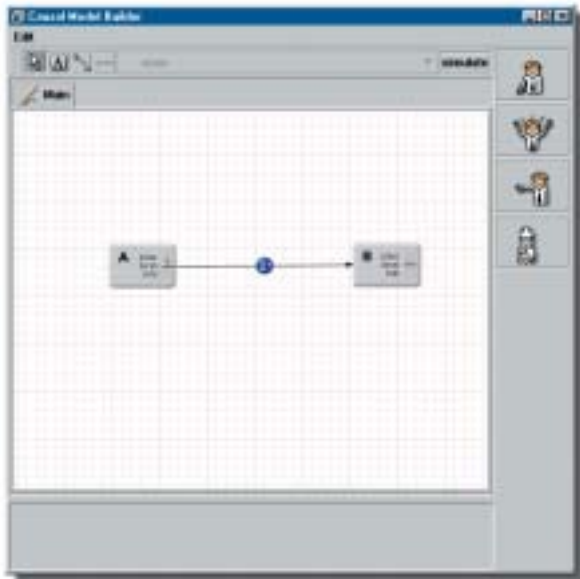
- First, there is the idea of early feedback and easy expression. This idea appealed to all experts. Although the causal model pad is expected to work only with small and simple causal models, the experts appreciated the easiness of expressing causal models. The close integration of model building and simulation is appreciated because of the short feedback it provides. This short feedback makes the effect of modelling decisions apparent at an early stage of development.
- Second, there is the educational value such a simplified environment can have. The causal model pad is perceived to have a high educational value concerning comprehending the causal primitives, or qualitative calculus, in qualitative reasoning. The qualitative calculus can easily be made insightful to novices to qualitative reasoning. Easy and early feedback adds up to this: learners can easily see the behaviour of their (intermediate) causal model.

A number of interesting ideas were mentioned. The causal model pad can be used to manage a central vocabulary of causal model primitives. The idea of a library of generic causal model primitives was mentioned. The idea of *causal design patterns* was mentioned. In object oriented programming, design patterns strongly influenced the way developers think about systems [33]. Causal design patterns can do the same for qualitative reasoning, and the causal model pad can play a role in this. To do this, the causal model pad must be able to easily communicate its content. This calls for import and export functionality. Besides the use in the MOBUM environment, experts mentioned that the causal model pad can have an educational value as a stand alone modelling tool. The causal model tool provides a modelling environment by itself. This tool can be used to teach learners about the qualitative reasoning domain.

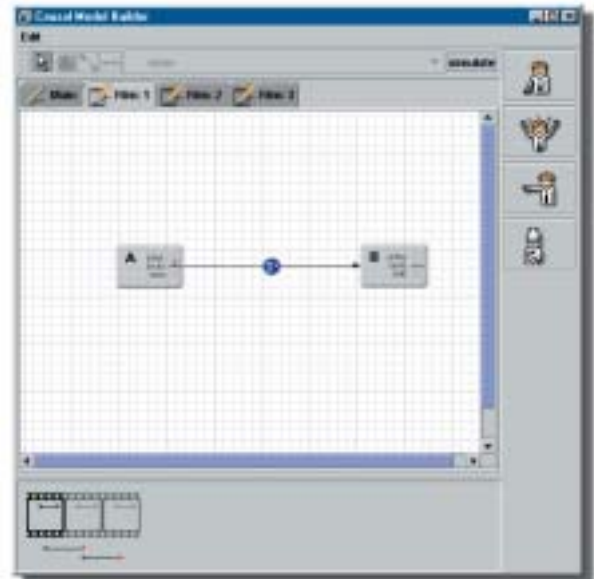
Major drawback of the causal model pad is its simplification: the easiness of expression is traded against expressiveness. Entities and quantity spaces are omitted, as well as the ability to create model fragments. This limits the modeler to very simple causal models. More complex causal models cannot be expressed in causal model pad.

To conclude, all tools are appreciated by the domain experts, although the behaviour envisioning was not appreciated unanimously. The high level of integration of the tools in the modelling environment contributed to the positive evaluation. Based on the evaluation by modelling experts, one can conclude that the two specification components do have an additional value in the model building environment. Furthermore, the communicative aspects of sketches is a feature of interest. Also, a library of causal model fragments is an idea that deserves attention.

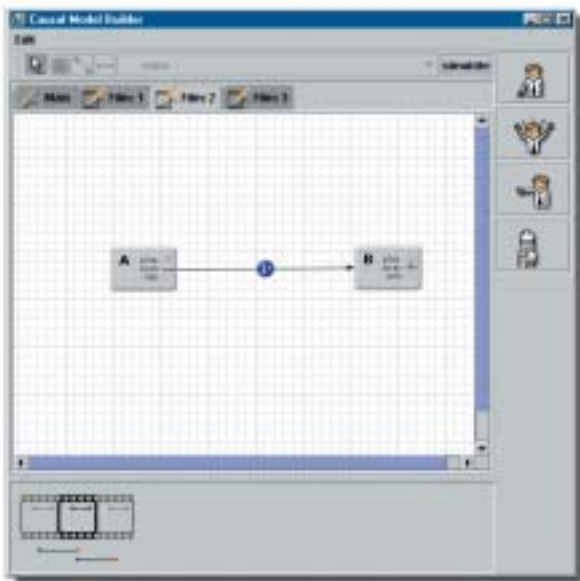
a



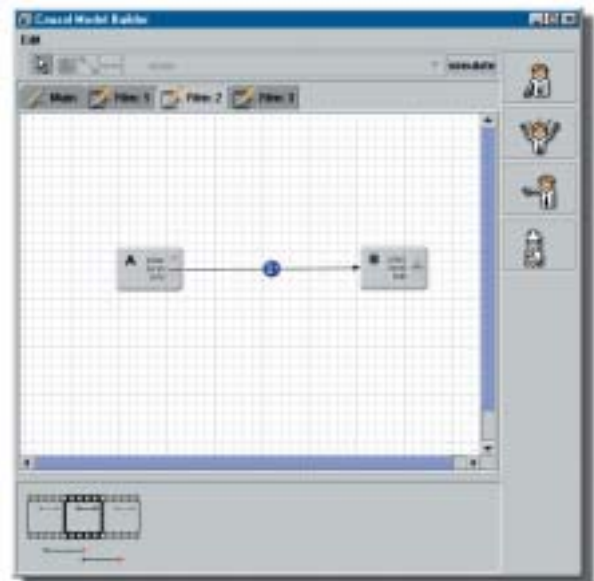
b



c



d

**Figure 3.9**

The causal model pad displaying a simple model. a) The model consisting of two quantities A and B. A has a positive influence on B and A is zero and going up. b) The first state displaying A rising. c) The second state: A is positive and B is rising up. d) The third state: B has become positive.

GarpApplet: The Output

4.1 Introduction: a GARP Simulation

As discussed in section 2.6.4, qualitative model construction consists of three tasks: model creation, behaviour prediction, and simulation inspection. In this chapter a tool for simulation inspection is discussed.

The GARP [14] simulator originally produced a text-based output. This seems sufficient for experts, but once the conception began to grow that this technology may be of educational value, the need for a more comprehensible output emerged. As discussed in section 2.6.5, visual representations of the output of a simulator can help to more easily understand and comprehend the output of a simulation.

The knowledge contained in the qualitative simulations is highly structured. This structure can be exploited to generate appropriate visualisations. If we take the behavioural knowledge contained in qualitative simulations, we can distinguish two different aspects of the simulation that need to be visualised:

The state graph The collection of states and their transitions in the current simulation. The behaviour of a state is manifested in the values of the quantities for that state. The transitions consist of the rules that make the states change. These rules consist of conditions, things that are true in the current state, and givens, things that will become true in the successive state.

The causal model The internal causal model of a state. The entities, relations, quantities, quantity spaces, and dependencies that capture the causal interpretation of the system behaviour in each state.

The inspection tool presented in this chapter uses different visualisations for each aspect of the simulation. These two aspects and the visualisations are discussed in sections 4.3 and 4.4. Besides visualising the content of a simulation, the tool provides means to run and control a simulation. The simulation starts by choosing a scenario, the user then can control the steps that the simulation makes from that point onwards. Simulation control is discussed in 4.2.

4.2 Simulation Control

A simulation starts of with a set of initial states. From that point on, the simulation can be walked through by executing different commands. These commands are, as described in [14] and displayed in figure 4.1, *terminate*, *order* and *close*. These three steps can be executed at once for a selection of states. Furthermore, an entire simulation can be run at once using the *full simulation* method.

Terminate Terminate the current selected states, selects the possible terminations that will apply for this state.

Order Order the current selected states, that is, order the current terminations in the state. This ordering consist of three steps: merge related, remove the obsolete, and assemble the cross-product.

Close Close the current selected states, actually, execute them. This will generate succeeding states, if any, or connect the transition with an existing state.

TOC Terminate, order, and close in one action: all the steps a state has to process to generate the terminations and the succeeding states they refer to, if any.

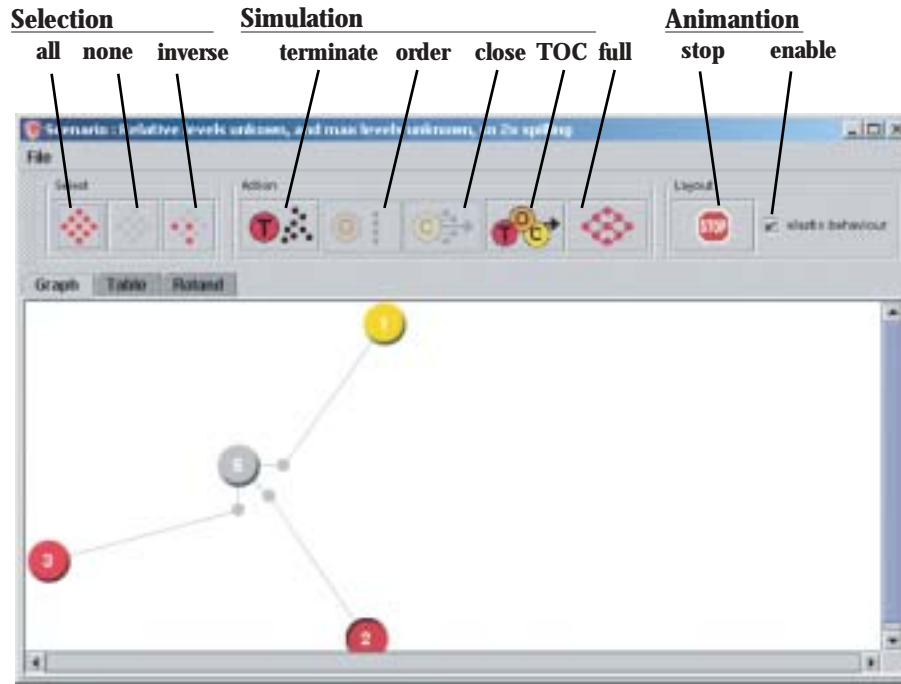


Figure 4.1

The simulation frame displaying the initial states of the u-tube simulation. The simulation control buttons are located in the toolbar.

Full simulation Do the full simulation at once.

The control buttons are accessible from the simulation frame. These buttons are not always enabled. The availability depends on the selection in the simulation view. The contents of the selection of states controls for the availability of the different control buttons. If for instance a terminated state is selected, all buttons except the terminate button are enabled, because the terminate button represents a function that does not apply to the current selection. The availability of the buttons always covers the smallest set of possibilities. If, for example 5 states are selected and only one is terminated, the terminated button will be disabled.

4.3 The State Graph

A simulation typically exists of an input system and one or more states. These states are connected by transitions. This structure is a graph. A graph consists of nodes (states in GARP terms) and edges (transitions in GARP terms). Nodes are connected to each other by their edges. The state graph does not have a hierarchical structure, there is no top node and the connections are not hierarchically structured or directed in one direction. This kind of graph is a unstructured and undirected graph, which is hard to visualise, see [19], and [37].

Kuipers [42] used a method that simply displays all the paths in a state graph. This means a state can have multiple instances, because they are displayed in each path in which they occur. When this method is applied to more complex simulations, lots of redundant information will be displayed. This kind of graphical representation is not optimal in the use of screen space. On the other hand, it does show the paths of the state graph insightful.

4.3.1 Visualisation

The state graph conveys different classes of information. What kind of information must the state graph visualisation make insightful? The requirements for an appropriate graph visualisation are defined as:

- a. Making the *paths* in the graph insightful. The connecting paths between two states depict the change of system behaviour that lead to a certain state.
- b. The *behaviour of the simulation* is conveyed in the values of the quantities in each state. This behaviour is a key aspect of the graph.
- c. The *overview of the structure* of the graph. Although graphs may be complex, there must be an overview of how states and groups of states connect.
- d. The *order* of the transitions and the *relative positions* of the states must be insightful. It must be clear what states precede and what states succeed a certain state. It must be easy to deduct whether a state is an end state in the graph, that is, a state that does not have any succeeding states.

One perfect method for visualising the state graph has not been found yet. To overcome this problem three alternative graph representations are used. One is a force-directed method for visualising the structure. The second is a heuristic approach, displaying the graph as a table. The third makes use of the temporal order of the simulation.

Multiple methods of displaying the state graph enable the user to inspect different aspects of the state structure, since different visualisations emphasise or suppress certain aspects of the state graph. Multiple representations enable the user to have multiple, supplementary, views on the same graph. Different features of the simulation representations facilitate in different search and indexing strategies. These representations are an instance of a *bridging representation*.

Although different representations are used for displaying the state graph, their basic graphical notation is similar. A state is visualised as a circle, coloured depending on its status. A transition is visualised as a black line connecting two states, with an arrow pointing in one direction, depicting the direction of the transition. The visualisation basics do not differ from any other graph visualisation in principle.

Force Directed Graph Visualisation: The Graph View

One of the most appealing methods is to view the graph as a *system of forces* acting on the states and transitions. By simulating the evolution of the system towards the minimal energy state, the system will organise its graph on the screen in an intuitive fashion. Force directed graph is a mathematical optimisation problem.

Advantages of *force-directed graphs*, or *spring layout boxes*, as described in [36] and [19], are the well balanced layout of undirected and unstructured graphs. The drawbacks of the algorithm are, first, its mathematical complexity. This complexity makes the algorithm perform poorly. It takes a considerable amount of processing power to reach a minimal energy state. The second drawback is the nature of the algorithm itself; the result of algorithm is unpredictable. Each run has a different result.

Both these problems are tackled by presenting the layout algorithm evolve as a live animation, that is, as a continuous animation, as presented by Huang [38] who uses this method for displaying relations in the WWW. In case of the first problem, this will enable the user to continue interacting with the interface while the calculations of the algorithm continue. In the case of the second problem, background simulation will make the user aware of the nature of the layout algorithm and its specific behaviour.

Visualisation method The basic method for *force-directed graphs* comes down to treating the states as nodes. The transitions in the simulation are viewed as edges, or in the case of force-directed graphs, as springs. These springs have a length. If the spring is longer than its normal length, it will have a pulling force on its two nodes towards each other. If the spring is shorter than its normal length it will have a force pushing the nodes away from each other. Figure 4.2 demonstrates these effects.

The second type of force that works on the states is electrical repulsion. All nodes try to maximise the distance from the rest of the states. This to prevent the nodes from cluttering. This goes especially for

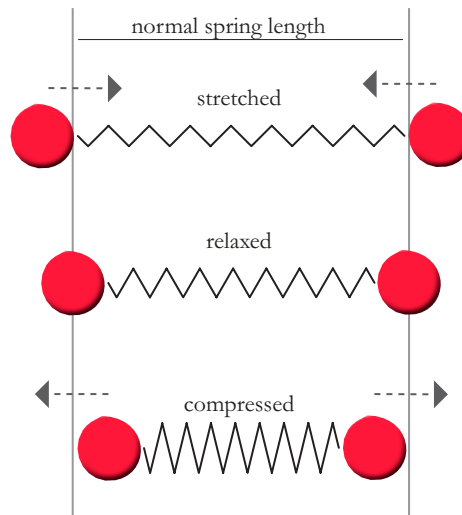


Figure 4.2

The basic workings of the force directed algorithm.

nodes that are not connected to each other by an edge. The force of repulsion is smaller than the spring force, otherwise the nodes would never stop moving away from each other. The forces combined end up in one direction and velocity for every node, moving the node to a minimal energy state. See figure 4.3 for an example of three nodes and two edges and the forces that govern their movement.

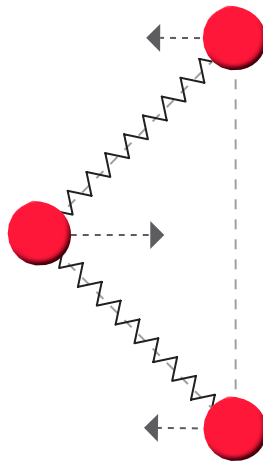
User interaction The visualisation enables the user to select one or more states by pressing the shift key while selecting the states. The organisation of the graph on the screen can be modified by the user. The state objects can be dragged to different locations, and the elastic algorithm will continue its search for the minimal energy state. The elastic behaviour can be disabled, if the user feels it does not behave appropriately. The transition objects can be selected, but not moved from their position relative to their from-state.

Discussion First, the paths of the simulation are visualised insightful, it is easy to deduct which paths connect to other states and what routes lead to a certain state (requirement a, section 4.3.1). The representation reads as a road map. Second, the structure of the connections in the graph are made insightful (requirement c). Third, the relative position of the states is made insightful (requirement d). The end states, the states which do not have any successor, are also easy to find.

Displaying the states as a force-directed graph has certain drawbacks. In case of high complexity, that is a high number of states (nodes) and transitions (edges) having multiple, unstructured relations, this results in a set of hard to unravel nodes. Such a complex graph is hard to comprehend. An example of such a situation is shown in 4.5. Second, and more important, is the lack of displaying the corresponding values of the states. The behaviour of the graph is thus not made insightful (requirement b).

The Table View

Displaying the behaviour of the system, that is the values the quantities have, is a crucial aspect of the simulation. To display the corresponding values, a visualisation concept taken from the original GARP [14] software is reused. Figure 4.6 displays the table view. It uses the x-axis to create a spatial link in order to connect the states to their values.

**Figure 4.3**

The different forces on 3 nodes (states) connected by two edges (transitions) in the force directed graph. First there are the springs pulling the states towards each other. Second is the electrical repulsion that pushes all the nodes away from each other..

Visualisation method The states of the graph are visualised as columns of a table. Each column represents a state of the simulation. The corresponding state is on top of each column. This table layout denotes a spatial meaning to the x-axis. The x-axis represents the different states as columns. This spacial meaning refers to behaviour of the states displayed on the lower half of figure 4.7. This spatial connection on the x-axis connects the states with their behaviour. This is supported by the use of shading. The two graphical panels are connected; if one panel is scrolled, the other adjusts its x-axis in order to keep the two corresponding.

Each row in the top panel represents a transition, as an arrow starting from the from-state, pointing towards the to-state of the transition. As in the top part of figure 4.7. The rows are sorted first ascending by from-state and second by the distance between the from-state and the to-state, ascending.

This behaviour is visualised by quantities having certain values. The quantities are visualised as rows, displaying the value for each state in the right column.

User interaction The states can be selected, but they cannot be moved from their position. A multiple selection can be made by holding down the shift key while selecting multiple states. Transitions can be selected in a similar fashion. A multiple selection can contain both states and transitions. Neither states nor transitions can be moved or dragged from their position.

Discussion This view makes the behaviour of the values in the system, requirement b, insightful. The transitions in the graph are easy to distinguish. Connecting the states with their values adds an important aspect of the system to the representation.

The drawback of this way of representing the state graph is in the strict organisation of the states. Only a horizontal ordering is possible. The transitions are ordered in the y-axis. This makes it difficult to keep track of the transitions and their connected states. In other words, their relative position is hard to comprehend (requirement d). The relatively large distance makes it hard to recognise and remember the set of connecting transitions in a path. The path and the structure of the graph is somewhat hidden and deducting the paths is a quite difficult task, depending strongly on the memory capabilities of the user. The paths are not made insightful (requirement a) and the overview of the structure of the graph is not made insightful (requirement c). Take for instance figure 4.8. Notice how difficult it is to recognise what paths connect state 2 and 14.

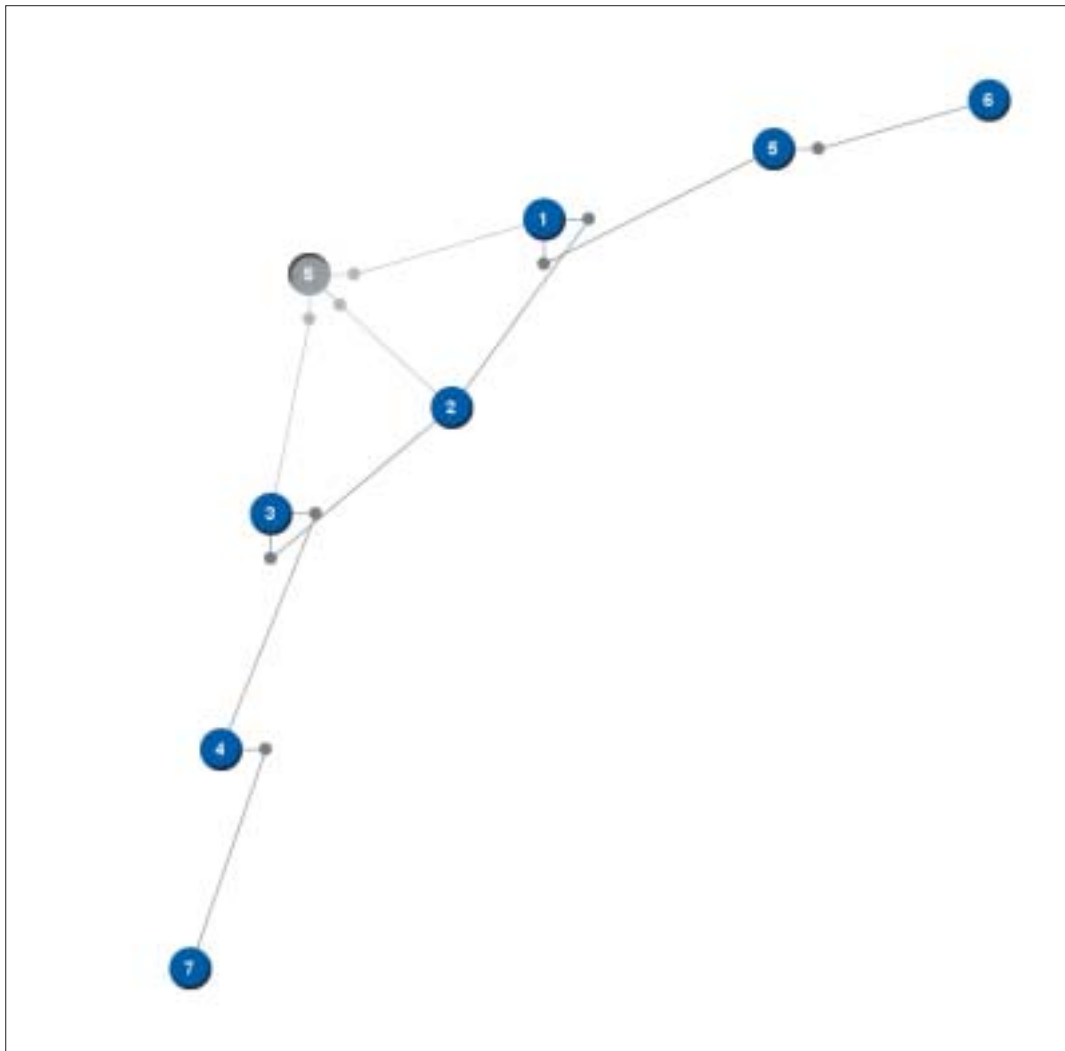


Figure 4.4

A picture from GarpApplet displaying the graph of a U-Tube simulation.

Notice how difficult it is to recognise the final states of the simulation. This information is hard to derive from this visualisation.

A more heuristic view: the Temporal Column View

This view is based on the heuristics derived from the temporal structure of simulations. A simulation always has a set of initial states. This set can be viewed as a current set, or *step* in the simulation. This current step can have a set of succeeding states. This set of succeeding states, if any, are the next step in the simulation. The discovery of new steps in the simulation continues as long as the current step has succeeding states, which are not allocated to any other step. States that belong to one step do not occur in any other step. A simulation consists of a finite set of steps. These steps represent the temporal succession of the simulation.

Visualisation method The layout of this method is a grid layout. Each set of current states, or step, is arranged in a column, starting with the initial set. Each succeeding set is placed in the adjacent column. The algorithm will attempt to put the succeeding state at the same height in the grid as the current state. This will minimise the distance between the two states. If this is not possible, the nearest available location

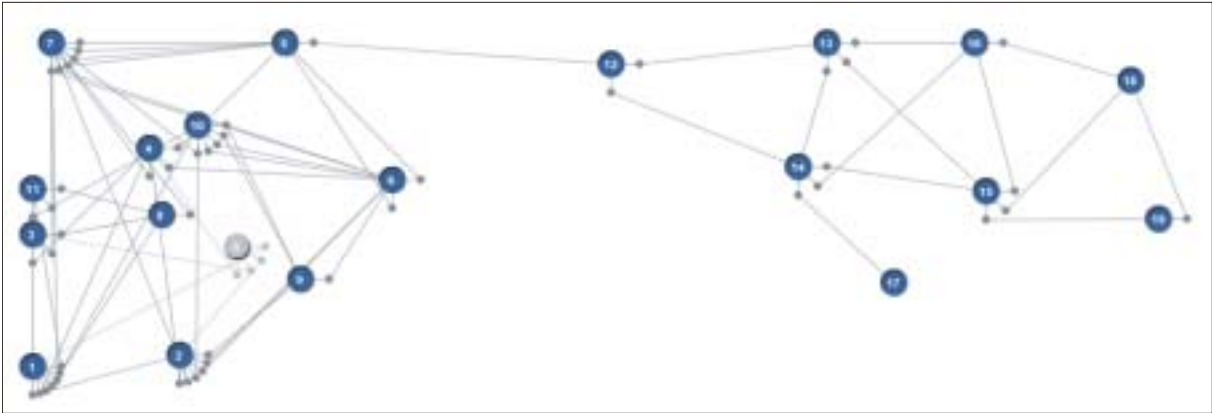


Figure 4.5
A screen-shot of one of the most complex models currently modelled in GARP.

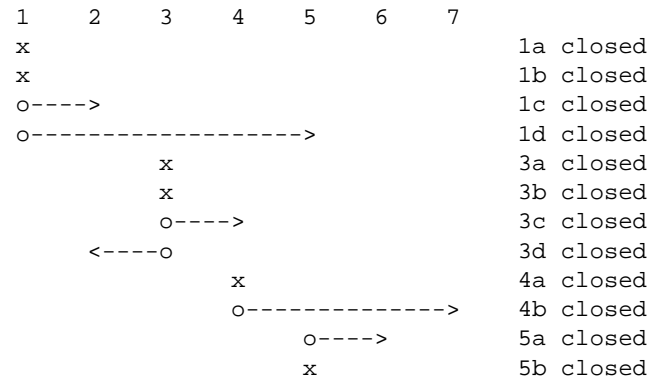


Figure 4.6
The U-Tube simulation by the original text-based GARP [14] interface.

on the succeeding column is assigned. If any succeeding state is already positioned, it will remain on its position. The connecting line will be drawn the other way around.

The representation uses different styles of arrows. Forward connections are straight; backward and sideward connections are arched. Colour coding is used for the transitions; once a state is selected the incoming transitions are coloured blue and the outgoing transitions are coloured red.

Figure 4.9 displays a temporal column view evolving over a three-step simulation. The initial states {1,2,3} are placed in the first column. Figure 4.9a displays this initial situation. The three initial states have four transitions {1-2, 3-2, 1-4, 3-5}, which result in the next step. In this step, two states {4,5} are new, state 2 already belongs to the initial set of states. Thus, the second column consist of two states {4, 5}. Note that the succeeding states are positioned on the same height as their preceding states. Figure 4.9b displays this step in the simulation. The two states {4,5} from the first step have two transitions {4-6, 5-7} which produce two new states {6,7}. The second step thus consists of those two states, positioned on the same height as their preceding state. Figure 4.9c displays the new column. States {6,7} do not have any transitions, so the final step of the simulation does not produce any new states. Figure 4.9d displays the final step of the simulation.

User interaction In the temporal view one or more states can be selected. They cannot be moved or dragged. Transitions cannot be selected.

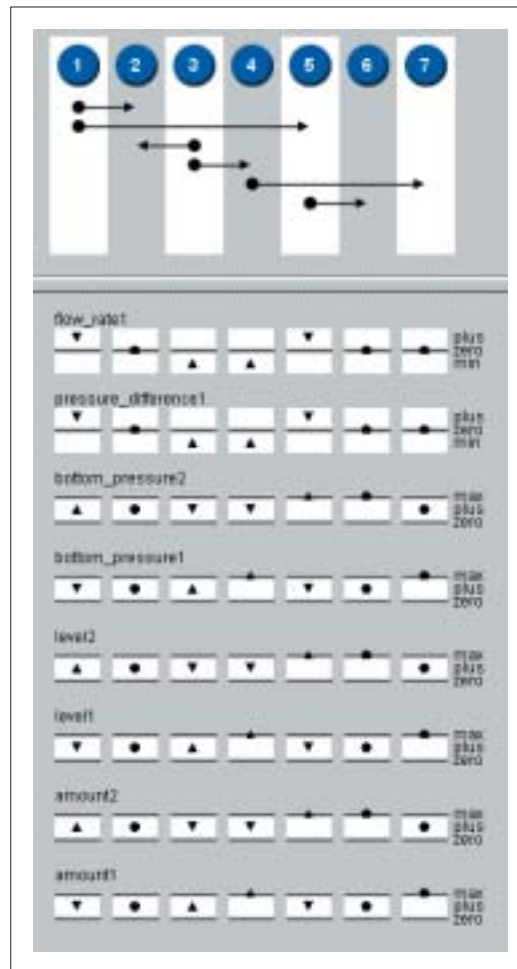


Figure 4.7

The table view on the graph on the top, with beneath that a portion of the values of that simulation. This picture is taken from the U-Tube model.

Discussion This heuristic view provides the user with temporal insights not provided by the other representations. The columns represent steps in the simulation, the x-axis represents the progress the simulation makes. These temporal insights make clear how branches proceed, how they end, and how they relate to other branches, information not displayed by the other two representations. The usage of different styles of arrows prevents the arrows from cluttering together and keeps them identifiable. The usage of colour contributes to this. Figure 4.10 displays the one of most complex simulations in the temporal column view. Summarising, the paths in the graph are made insightful (requirement a). The structure of the graph is made insightful (requirement c). The relative position of the states is easy to deduce from this visualisation (requirement d).

Major drawbacks are: first, the drawbacks of a the grid based layout. Such a layout uses considerable more space and does not respond as optimal in its spatial efficiency. Second, it does not have meaningful spaces for displaying values, making it impossible to graphically connect the states to their values as the table view does. This visualisation does not meet requirement b since it does not make the simulation's behaviour insightful.

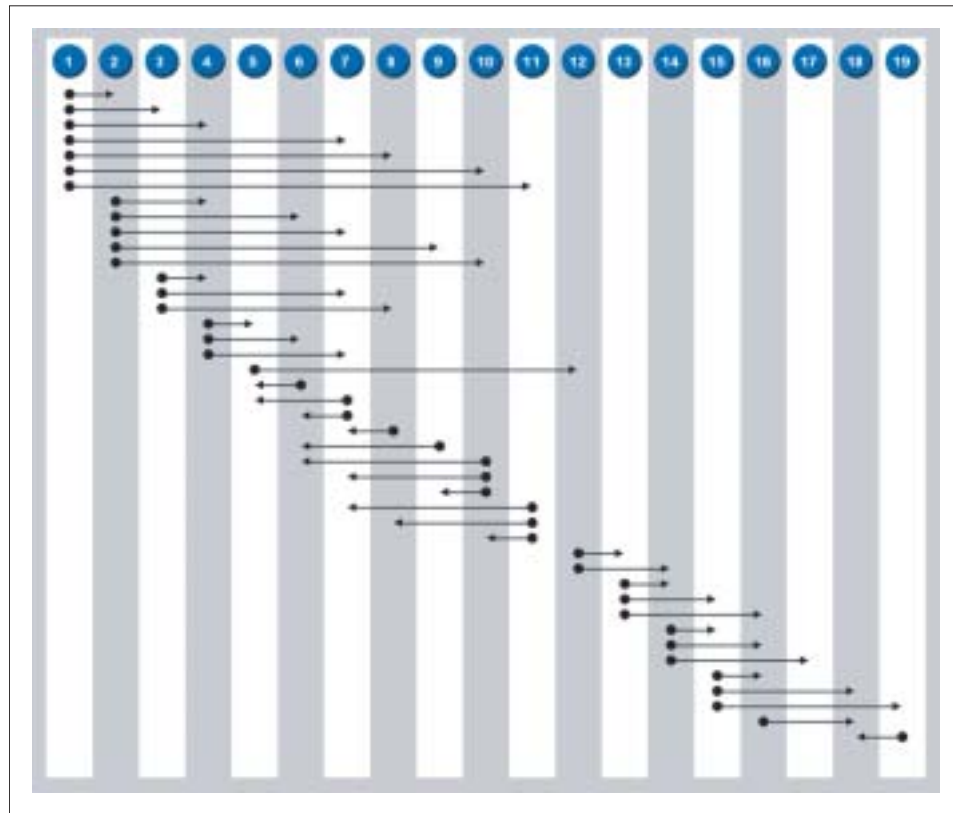


Figure 4.8

The table view showing on of the scenario's of the Cerrado model, one of the most complex simulation currently available.

4.3.2 The Transitions

The transition rules are rules that apply to the current state. They can result in a new state. This depends on the validness of the product of the rules. Rules typically consist of a conditional part and a consequential part. The conditions are the model parts that need to be present in order for the rule to be true. Consequential parts are parts that are a product of the rule and become true.

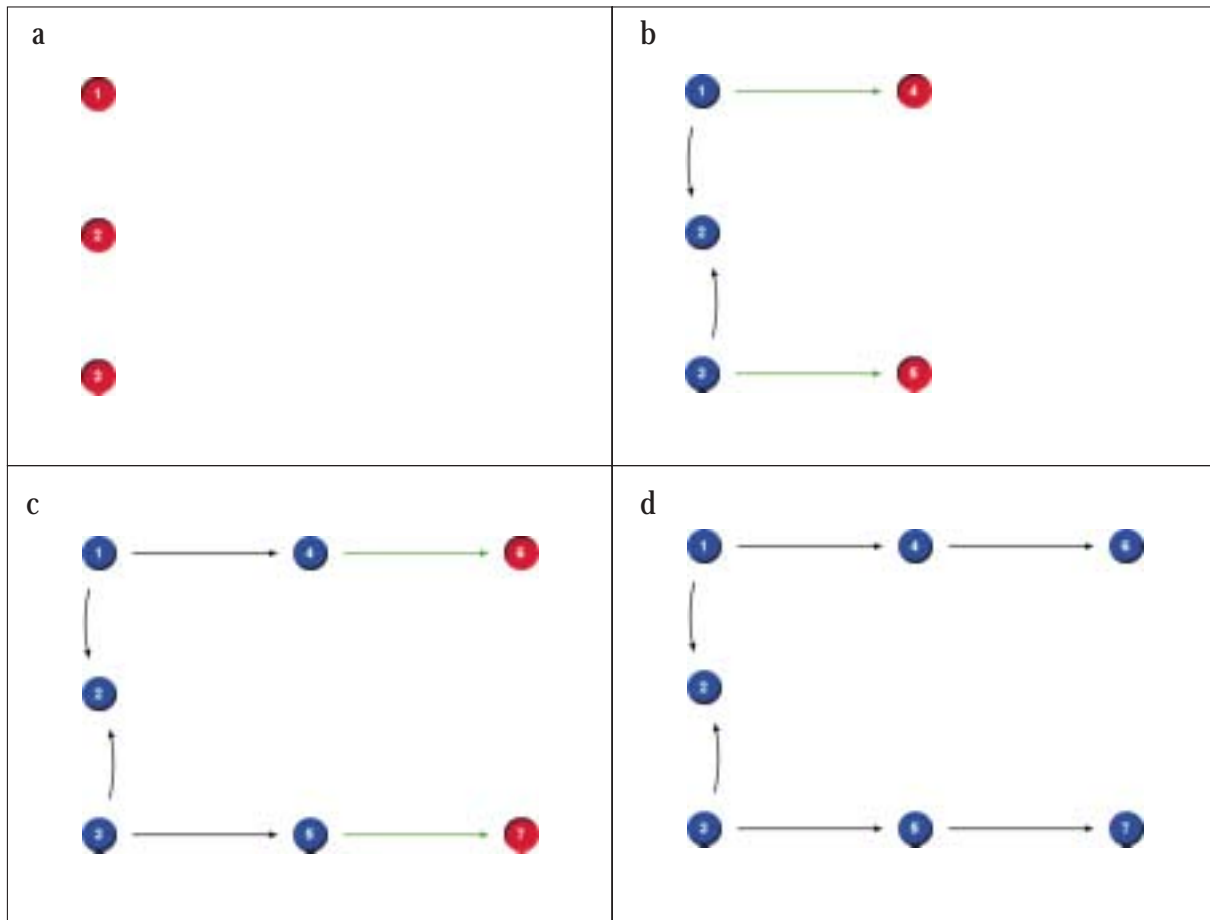
The transitions use both a textual and graphical representation. The textual representation reflects the notation used by the simulator itself. It uses colour coding in order to highlight the important segments. Red is used for conditional model parts, blue is used for depicting the consequence. The graphical representation resembles the presentation of the causal model, as described in section 4.4. It only differs in respect of the colour coding, which is similar to the text encoding.

4.4 The Causal Model

The causal model consists of entities and their quantities. These quantities have a quantity space. A quantity space consists of a set of values. Dependencies can exist between quantities, quantity spaces and values. This structure can be exploited to make effective visualisations.

4.4.1 Visualisation

What kind of information must the causal model visualisation make insightful? The requirements for a suitable visualisation are:

**Figure 4.9**

A u-tube graph visualised in a temporal column view.

- The visualisation must make the *primitives* of the causal model insightful. These are the entities, structural relations, quantities, quantity spaces, and dependencies.
- The visualisation must be able to cope with *complex causal models*. The causal model can potentially contain large amounts of primitives. These primitives are hard to visualise in one single visualisation.
- The visualisation must be highly *adjustable*. The user must be able to adjust the visualisation to its needs. Different aspects of the causal model must be isolated in order to investigate the structure of the causal model in detail.

4.4.2 Basic Concepts

In this section the design and rationale underlying the design choices are elaborated. The causal model has a distinct structure that can be exploited to create insightful visualisations. In this section the basic visualisation building blocks are discussed. The behaviour of the visualisation as an anticipation to the actions of the user is discussed in section 4.4.3.

Entities can have multiple quantities and quantities always belong to one entity. Figure 4.11 depicts this container-like structure. Entities act as *visual containers* [11] for their quantities. The entity *man* contains three entities: *universal*, *zero1*, *energy1*, and *height1*. The one-to-many relationship of entities and the quantities are visualised by using the visual container metaphor.

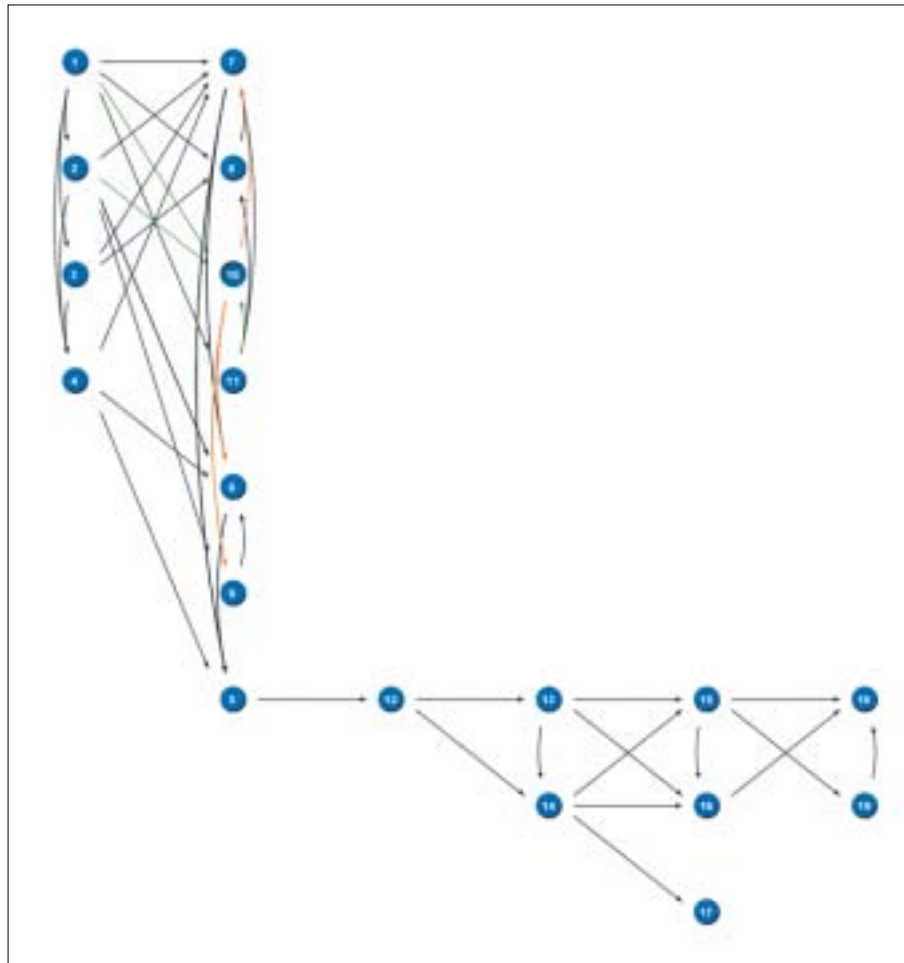


Figure 4.10

The temporal column view. Each column represents one step in the simulation. This figure displays the same simulation and scenario as figure 4.8 and 4.5 do. State 10 is selected.

Quantities are visualised as text with a line underneath. The line starts off with a dot. The line is used to visually connect the quantity space, and separate the quantity from the connected quantity space.

Quantity spaces and their quantities have an one-to-one relation. Each quantity always has one quantity space. This is visualised by visually connecting the quantity space to their quantities. The quantity space consists of successive points and intervals. Points are visualised as dots, intervals are visualised as vertical lines.

Dependencies and **inequalities** connect the two items they refer to. Each type has a type-specific icon in the centre of the connecting line. If multiple dependencies exist between the same two items, they are arched in order to keep their icons visible. For instance, in figure 4.11, the $I+$ and $I-$ dependencies between `height1` and `energy1` are arched in order to prevent them to overlap.

Relations between entities are visualised as lines. The arrow depicts the direction of the relation and a textual label displays the name of the relation. Figure 4.11 displays the `connected to` relation between `man1` and `elastic1`.

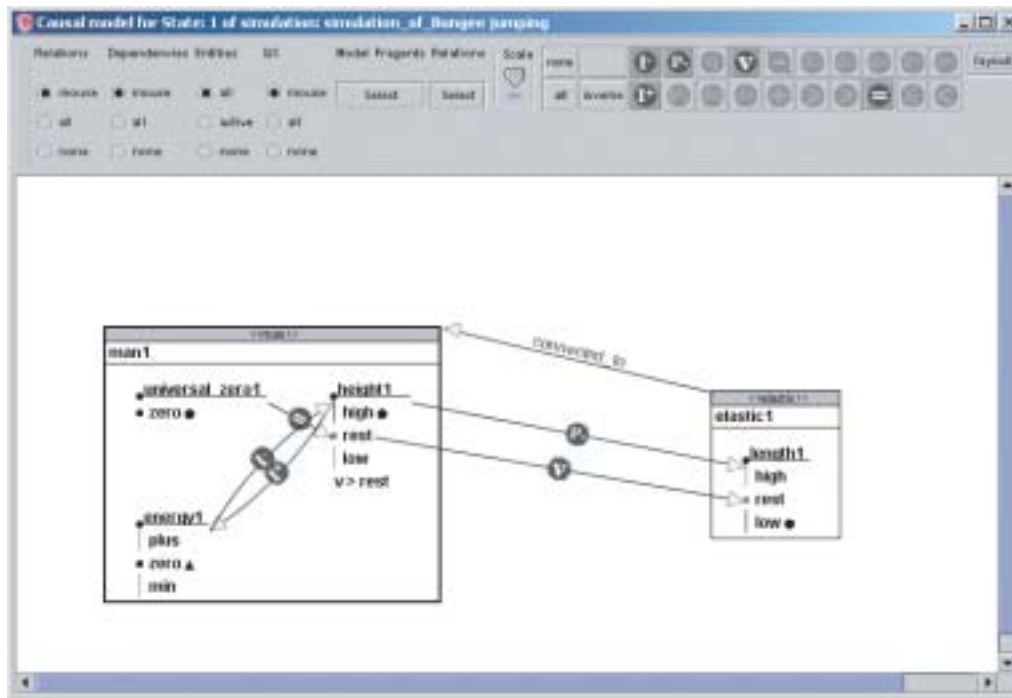


Figure 4.11

The causal model inspection tool visualising a causal model.

4.4.3 Complexity

The challenge of visualising the causal model lies in how to deal with large causal models. In other words, dealing with complexity is a key issue. Causal models usually contain large amounts of information, that, if presented all at once, is hard to comprehend. There are various approaches to control this complexity. The causal model visualisation uses the following concepts to deal with this complexity.

1. *Hiding and highlighting* [41] can put the emphasis on certain aspects while suppressing other aspects.
2. *Graphical Zooming* [36] can be employed to deliver overview or inspection of details.
3. *Semantic fisheye* [63] is a focus+context [32] layout technique, controlled by the mouse pointer, that lets the user control the level of detail in the visualisation.

Hiding and Highlighting

The concept of hiding and highlighting is applied in the causal model visualisation in two ways.

First, the user is able to hide or activate certain classes of primitives of the visualisation. This method simply hides or activates an entire class of causal model parts. This can be quite informative while searching for a specific type. Figures 4.12, 4.13, and 4.14 are examples of entire classes of primitives that are hidden or activated in the visualisation. A more elaborate example of hiding is displayed in figure 4.15. In this example, a specific type of dependency (P+) is activated, while all other types are hidden. This enables the user to simplify the visualisation and concentrate on a specific set of dependencies.

Second, hiding and highlighting is applied when certain parts of the causal mode need to be grouped. This is typically used when certain model fragments, structures that would otherwise not be visible, are inspected. Figure 4.16 depicts an active model fragment.

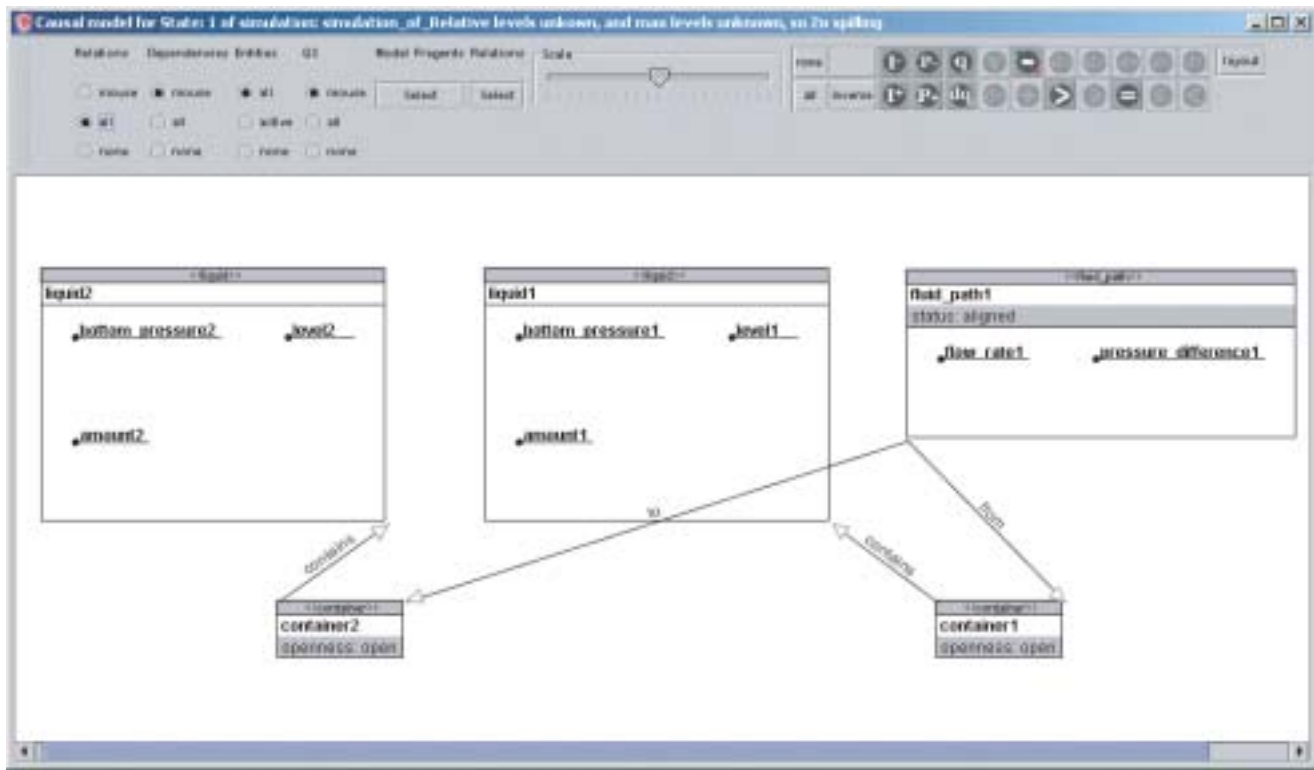


Figure 4.12

The causal model displaying the U-Tube with all the relations activated.

Zooming

The entire visualisation can be enlarged or reduced in order to get more detail or overview. The user can control this scaling by a slide-bar. Scroll-bars on the side of the screen control the panning of the view. To zoom the visualisation is simply scaled. Zooming out will make the objects on the screen smaller, zooming in will enlarge them.

Semantic Fisheye

The mouse-pointer controls a process that can be described as a *semantical fisheye*, a focus+context layout technique, controlled by the mouse pointer. By hiding, highlighting, and grouping parts of the visualisation, the level of detail is controlled. The mouse-pointer controls this process. The basic idea is that of a fisheye view, but the enlargement is not geographical, the physical properties objects are not enlarged. Instead, the level of semantical detail of the visualisation is enlarged. In other words, the visualisation becomes more specific once the mouse pointer hovers a certain area, or clicks a specific object. The visualisation expands its level of detail. The hovering of mouse pointer above the causal model parts will expand them into more complex representations. A semantical magnifier surfaces the visualisation. The user can browse and navigate the causal model by moving the mouse pointer over the visualisation and clicking certain highlighted objects. Figure 4.17 illuminates this behaviour. The entity in figure 4.17a displays an entity in its normal state. The entity shows its quantities. Figure 4.17b displays the entity while the mouse hovers the graphical object. The relation with another entity (which is currently not visible) is displayed. This is the first increase in detail. The mouse pointer changes into a hand shaped pointer to indicate the possibility to click the current entity object. Clicking the entity object will expand it and bring it to a state as displayed in figure 4.17c. The entity object is expanded and shows its quantities in a different layout. This different layout accounts for the expansion of the quantity representations. This expansion of the quantity

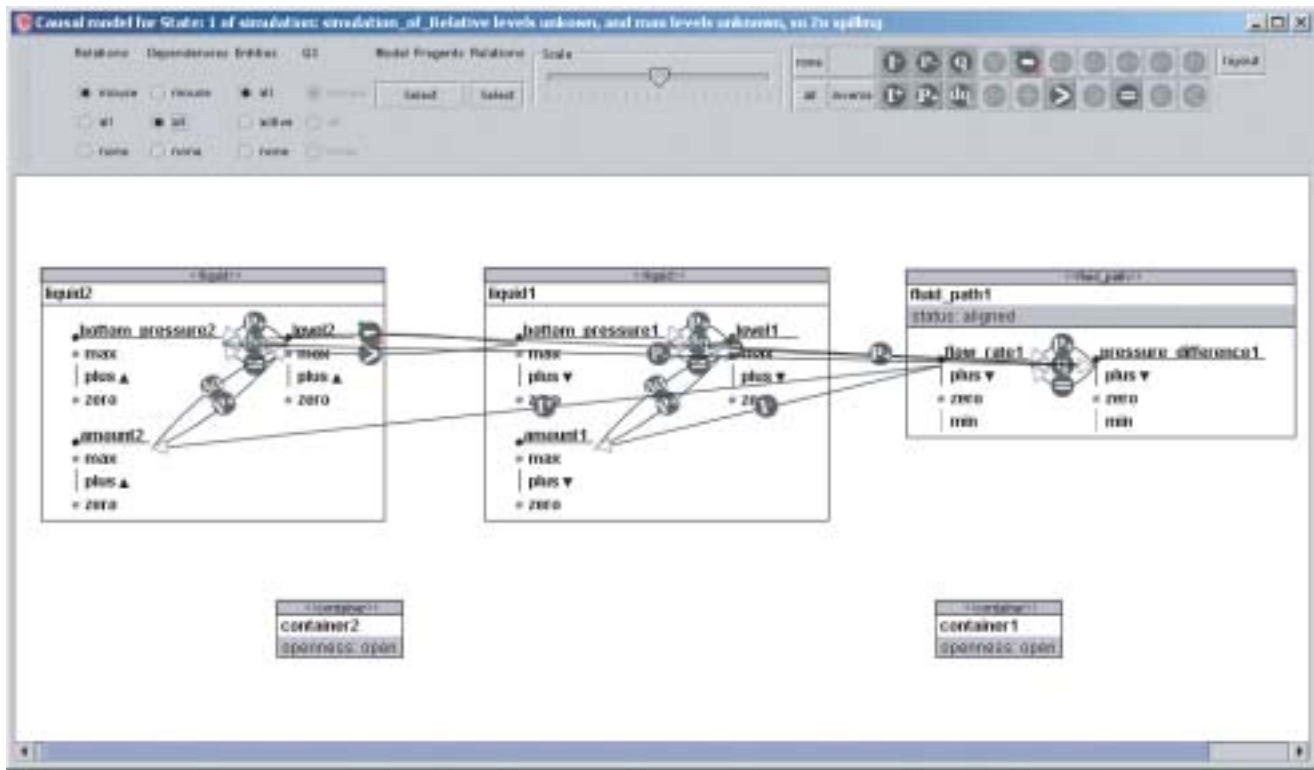


Figure 4.13

The causal model displaying the U-Tube with all the dependencies activated.

representation will occur when the mouse hovers the quantity. The quantity will then display its values and the dependencies connected to them. The connected quantities will also expand, and, if necessary, the owning entities of these quantities also. Here the interaction does not end. The values can be hovered also, and disclose their dependencies.

Only the clicking changes the representation semi-permanent. The quantities will un-expand once the mouse pointer leaves their area. The entities will un-expand by clicking their area again.

Figure 4.18 displays one of the most complex causal models currently available. Here the semantic fisheye proves its use. Such complex causal structures are difficult to comprehend. This layout algorithm supports the exploring of the structure by zooming in on the structures of interest.

4.4.4 Layout Algorithms

There are two layout algorithms active in the causal model visualisation. The first prevents the user from putting objects on top of each other. If an object is dragged on top of another, the bottom most object is moved until they do not overlap anymore. If moving the bottom most object is impossible, the dragged object will be moved instead. This algorithm is continuously active.

The second algorithm arranges the object on the screen in such manner that they are organised insightful to the user. The objects are sorted by complexity, starting with the most complex and ending with the least complex. Then, they are placed on a grid. This algorithm is only applied initially, but can be reapplied at any time.

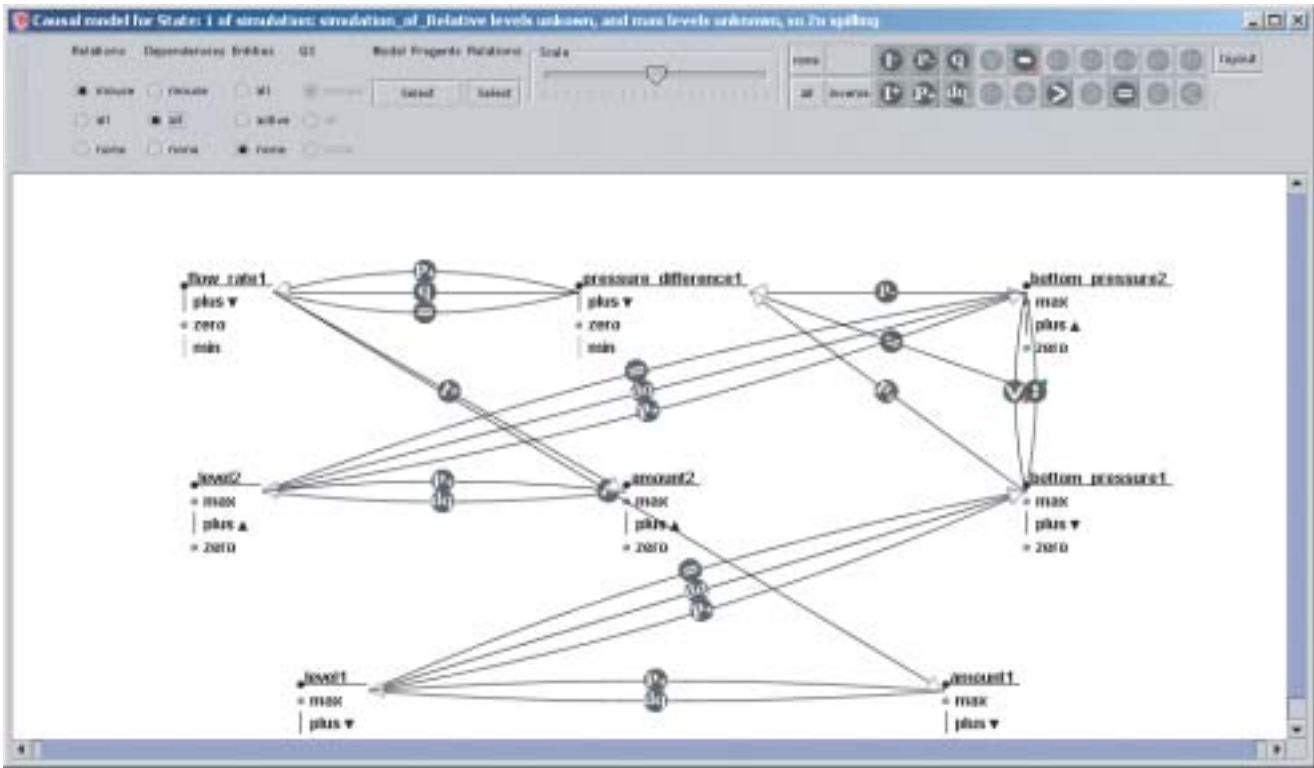


Figure 4.14
The causal model displaying the U-Tube with all the dependencies activated and the entities disabled.

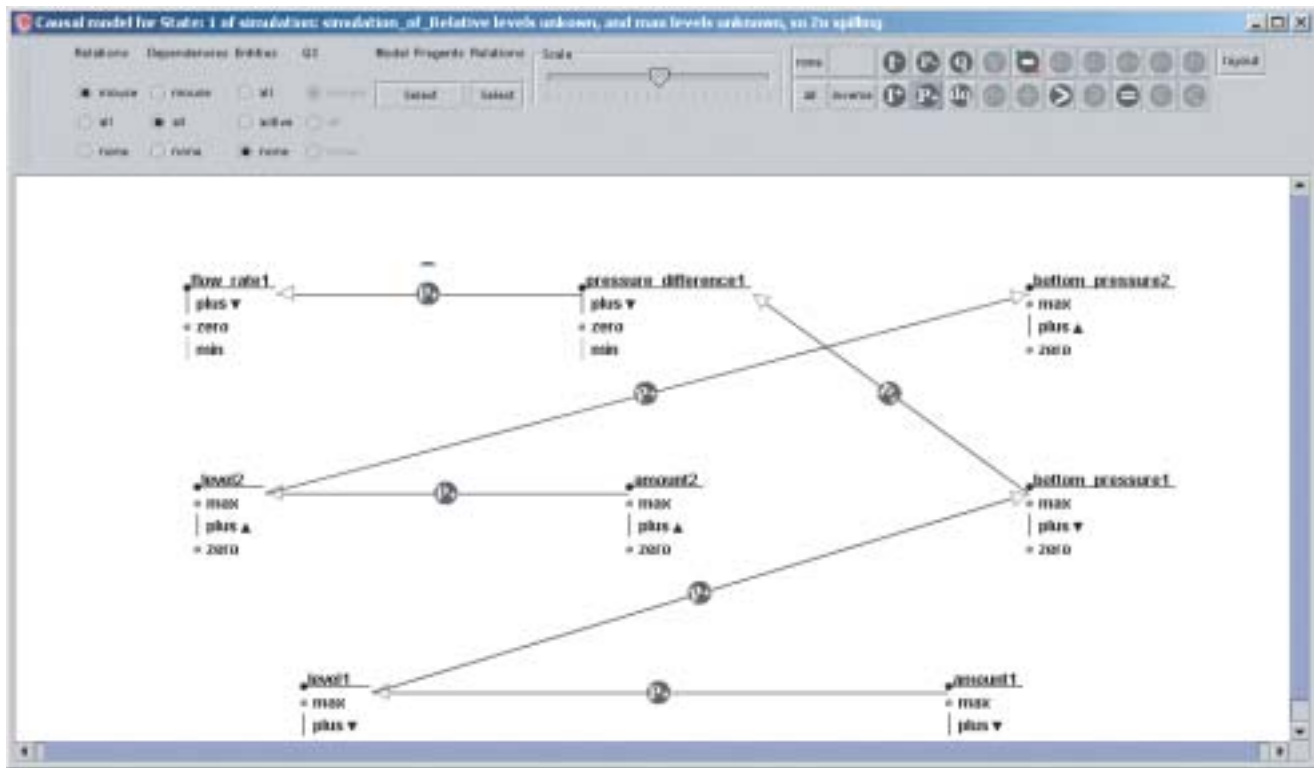


Figure 4.15

The causal model displaying the U-Tube with all the dependencies activated, the entities disabled, and only the P+ dependency enabled.

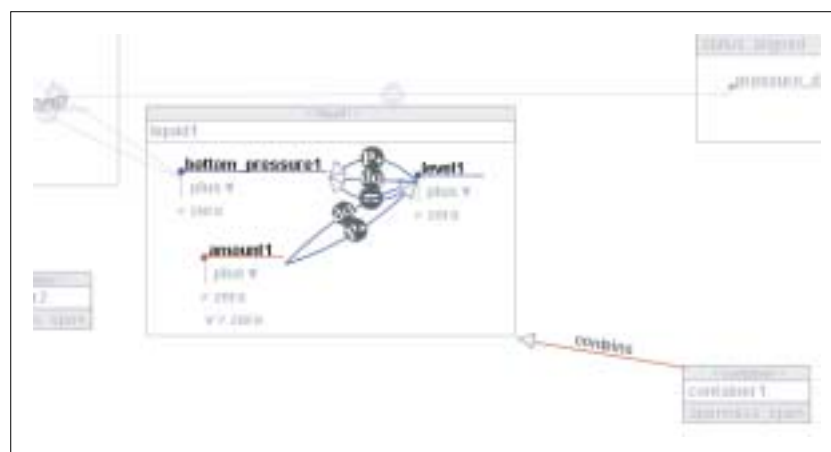
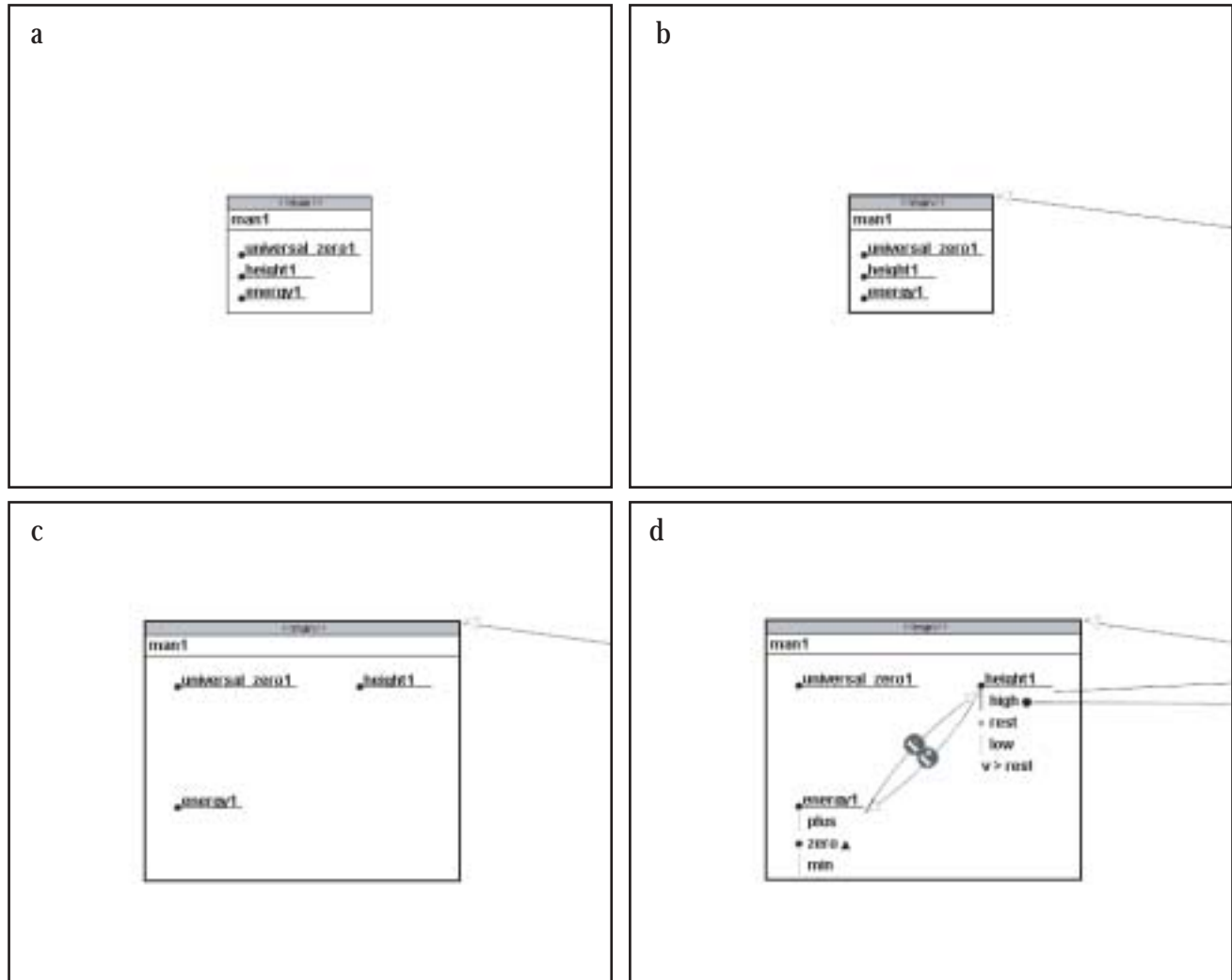


Figure 4.16

The causal model displaying a highlighted model fragment.

**Figure 4.17**

The behaviour of the causal model visualisation. a) the entity without any mouse pointer hovering b) the entity highlighted, the mouse pointer has entered the entity c) the mouse pointer has clicked the entity, it has expanded. d) the mouse pointer hovering the quantity, revealing its values and dependencies.

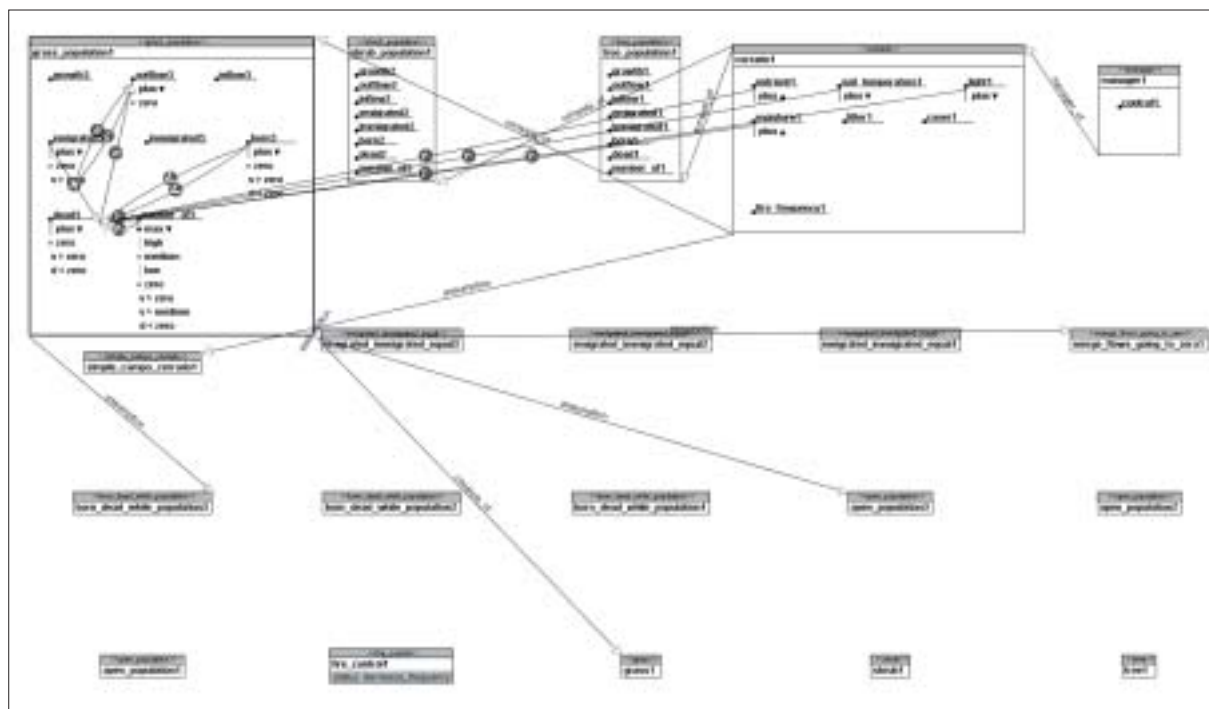


Figure 4.18

The causal model of a state from the cerrado model.

5.1 Introduction: Types of Support

The modelling process consists of three tasks; model construction, behaviour prediction and simulation inspection. The MOBUM environment is concerned with model construction and simulation inspection. Chapter 3 discusses how model construction can be facilitated by tools that convey different aspects of the model. In this chapter the support in the MOBUM environment is discussed. This support ranges from providing the right user interaction to intelligent agents that act as model building advisers or facilitate the model building process by integrating the various tools. We will make a distinction between three types of support. First there is inherent support, second, there is static support and third, there is dynamic support.

- *Inherent support* attempts to prevent misconception and errors by the design of the graphical user interface (GUI) and the user interaction. The interface can be used to explain and demystify certain behaviour. The user learns the aspects of a system by using it. A GUI should make clear what actions are possible and what actions are not.
- *Static support*, is support that is invariable and represented in only one form. Both dynamic and static support follow after a kind of *breakdown*. A breakdown occurs when a user is not able to fulfill a task for a certain reason. Static support, commonly known as help, provides the user with support information from static media. These media provide the user with textual information, combined with images and hyperlinks. The static support often describe the general use and common questions of the software.
- *Dynamic support* refers to intelligent support that dynamically anticipates on the user's actions and the system's state. This kind of support is variable and generated on the basis of the context. Dynamic support differs from static support in its nature. Dynamic support is related to the contents of the users work. Dynamic support reflects on the work of the user by providing for advice. To provide advice, the support system needs to be capable of reasoning.

These different types of support are discussed in the following sections.

5.2 Inherent Support

The MOBUM [5] modelling environment is the successor of the HOMER [39] building environment. The evaluation [6] of the HOMER environment brought up a number of GUI and user interaction concerns. These concerns have inspired the development of MOBUM. Preece [49] states that there are three major aspects that must be kept in mind for building effective GUI's:

1. The state of the system must be visible.
2. The system needs to make clear what kinds of interactions are possible.
3. The feedback of the system is of importance.

In our view, the GUI and user interaction of an environment plays an important role in supporting the model building process. The physical, or graphical, aspects, as well as the behavioural, or interactional, aspects of an environment can guide and scaffold the modeler in the model building task. These aspects of an environment can act as supportive factors. The graphical and interactional design of a system should depict its use, and thereby support the modeler by making clear what actions are available, and what actions are not. The behaviour interaction of the system should, optimally, also depict *why* certain actions are possible in one situation, and not in the other. An example demystifies this idea. If there is a dialog for creating a new entity, the action of creating a new entity requires a name to be provided. To enforce the user to fill in a name, the dialog disables the OK button until a name is entered. This not only forces the modeler to provide a name for the entity, but the dialog also informs the user about the mandatory nature of the name of the entity.

5.2.1 HOMER vs MOBUM

As mentioned, MOBUM's GUI and user interaction design is inspired by the evaluation of HOMER. For that reason, the MOBUM's design rationale is discussed in conformance with HOMER's design.

Error Prevention

In respect to preventing errors in the model building environment in HOMER and MOBUM there are two strategies that need to be discussed. First, there is the strategy of disabling options if the required conditions are not met. This prevents the user from making errors and, ideally, informs the user about the required conditions. These required conditions must be easy to discover. This strategy thus only works with simple conditions, like the name field example discussed before. The second strategy is to inform the user about the erroneous behaviour, explain the problem and allow the user to correct the error. This strategy displays an error message which explains the cause of the problem. This method is suitable for error classes with a more complex nature.

Both HOMER and MOBUM use the two strategies. Our argument is that first, MOBUM deploys them more consistently and second, MOBUM uses them more in conformance with the nature of the problem at hand.

In a number of cases, HOMER uses the strategy of disabling options. Figure 5.1 displays how HOMER uses this strategy to prevent the user from removing elements that cannot be removed from the system. The MZP quantity space is required in every model and cannot be removed. In other cases, HOMER chooses not to disable the erroneous option and inform the user about the erroneous behaviour. Figure 5.2 shows HOMER displaying an error message. HOMER is inconsistent in using the two strategies. There is no consistency in the strategies applied.

MOBUM attempts to apply the restrictive strategy in situations where the requirements of the conditions are easy to discover. An example of this behaviour is displayed in figure 5.3. In more complex situations, the MOBUM modelling environment will inform the user about the erroneous behaviour. Figure 5.4 displays an error dialog that explains why an entity cannot be removed from the structural hierarchy.

Availability of Options

The user interface should clearly display the possible actions in the state it is in. We argue that, in this case, more is better. The modeler must be aware of the possible actions and must be able to select these options in multiple ways. The latter will allow for individual interaction styles and increase the chance of discovery. Redundant methods of interaction is an inherent method for supporting the model building task. MOBUM uses three different methods for selecting actions. Most salient is the toolbar on the side of the screen. Their constant visibility contributes to the awareness of the possible actions: the availability of the buttons in the toolbar are adjusted to the selection. Drop down menus on top of the screen convey the same set of actions. Finally, context menus give access to the available set of actions. Figure 5.5 displays the three different methods for selecting actions. HOMER solely uses drop down menus for this purpose. HOMER does use context menus, but these menus do not offer all actions provided by the drop down menus. Not all actions

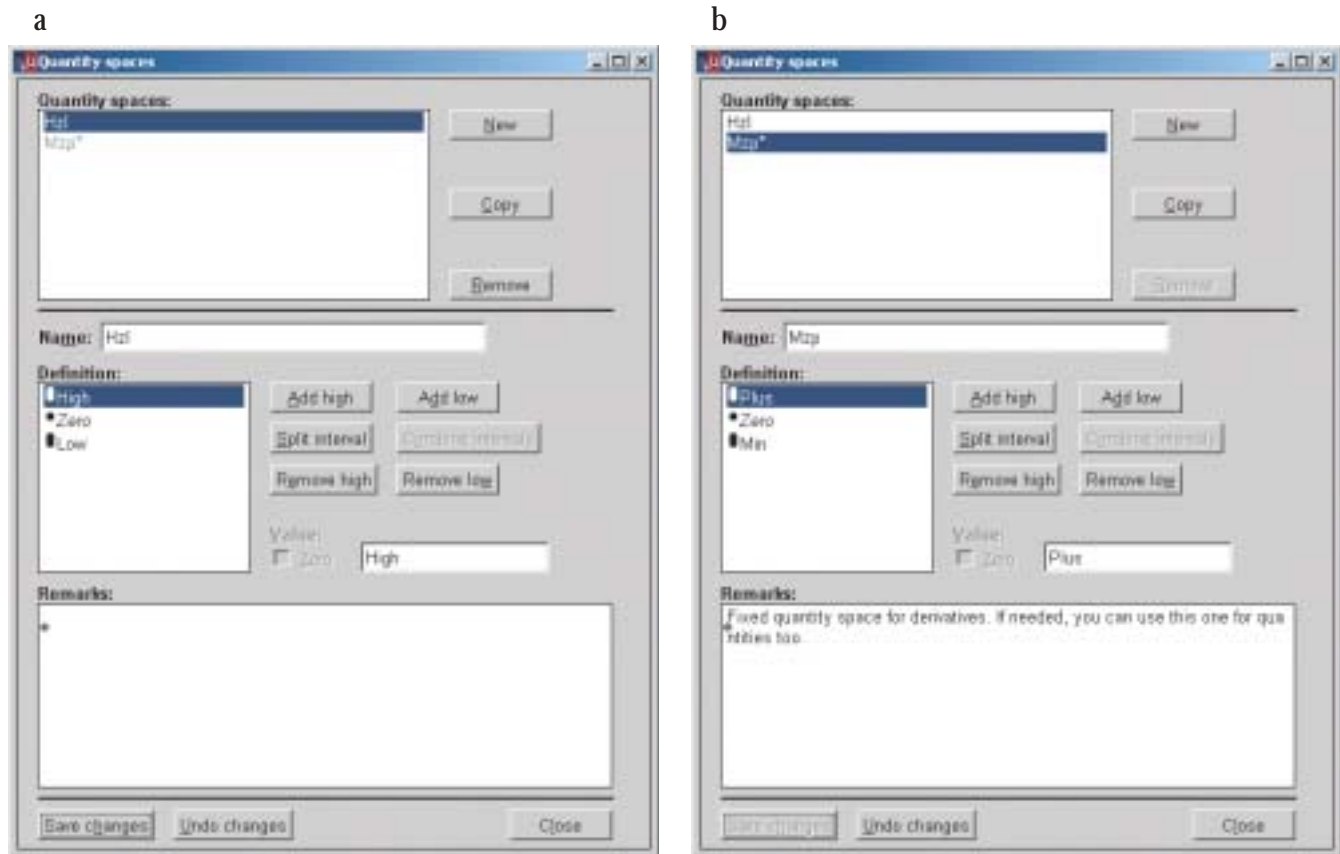


Figure 5.1
HOMER disabling the remove option to prevent the user from removing an essential model part.

available from the drop down menus are available from the context menu. The user does not immediately become aware of the possible actions since the adjustment of the selection is not manifested in a directly perceivable manner. The changes in availability of the actions is not visible while changing the selection. This makes the discovery of the available options a more complex task.

5.3 Avatars

Inherent support manifests itself in the design of the GUI and the user interaction and implicitly supports the model building task. This kind of support does not come to the modeler in an explicit form. On the other hand, static and dynamic support do have an explicit graphical representation. These kinds of support have a different nature: they both present a dialog to the user that presents information, and thus have an explicit manifestation in the user interface.

In the MOBUM environment, six personalities, or avatars [17], are employed to create a kind of personality for each type of dynamic and static support. Each avatar fulfills a task or provides a specific type of information. Each avatar has its specific appearance which matches the task they perform or information they provide. Mobum is equipped with five construction components, named builders, and two specification components, the SWAN sketchpad and the Causal Model Pad. Each of those seven components in the MOBUM modelling environment is equipped with its own, specific, version of the avatar. Figure 5.6 displays the six types of avatars. The first four avatars provide static support and are discussed in section 5.4, the latter two avatars provide dynamic support and are discussed in section 5.5.

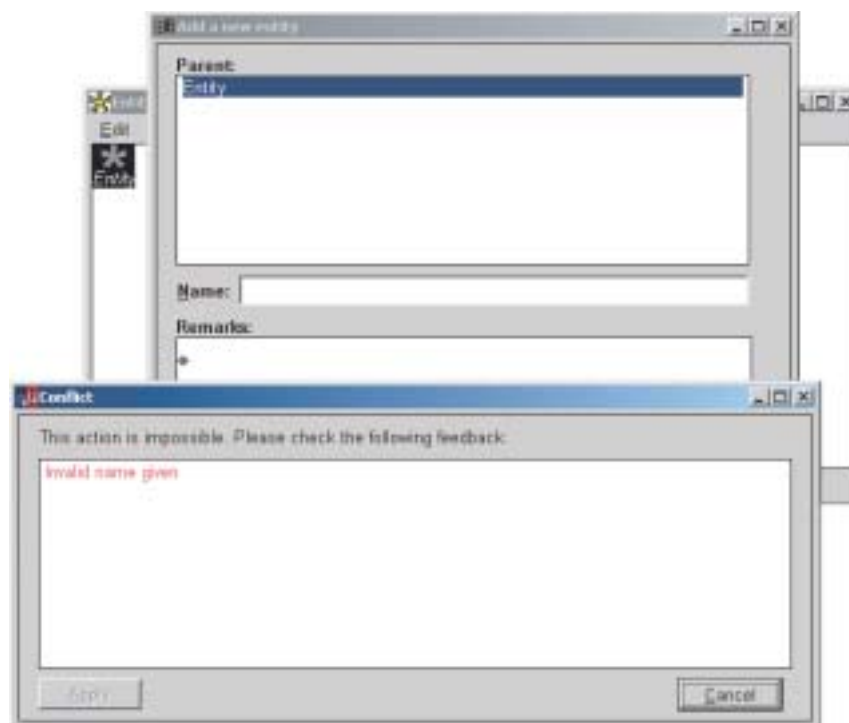


Figure 5.2

The HOMER model building environment displaying an error message.

5.4 Static Support

Static support provides the model builder with media such as text and images in order to answer a question or solve a problem. The four static avatars are: *What is?*, *How to create?*, *Curriculum planner* and *Global help*.

What is? This avatar provides help on model building concepts in the local context of a modelling component. Each component is equipped with its own version of the *What is?* avatar. This avatar provides a description of the model building concepts used in the specific construction components, and describes the conceptual aspects of the component. Figure 5.7 displays the complete text provided by the *what is?* avatar.

How to create? This avatar provides procedural help. In what order modelling steps should be performed and what actions to perform to reach a certain goal. Each specification components has a set of tools that constructs, manipulates, and removes the contents of the tool. The *How to create?* avatar provides the user with this procedural knowledge.

For example, the *How to create?* avatar for the Structure Builder provides the modeler with information on how to create attributes, entities and structural relations. The sequence of actions is described, as detailed as each field in each dialog that is necessary to fulfil the task.

Curriculum planner This avatar is currently not implemented. In the future, it should provide help on the curriculum. This avatar is specifically intended for educational purposes: students will consult this avatar to get information related to their assignments.

Global help This avatar provides the user with general modelling help. It has answers for questions like what qualitative modelling actually is. It provides an answer to the question: *How to build a*

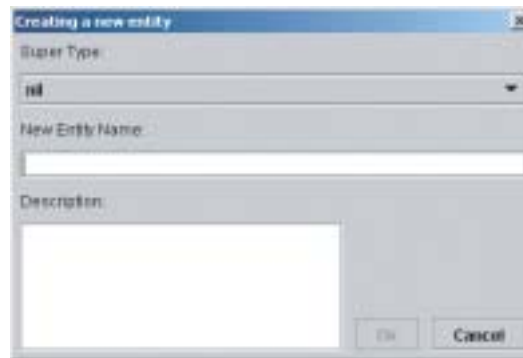


Figure 5.3

A dialog from MOBUM with a disabled OK button. The OK button is disabled because the name field is empty.



Figure 5.4

An error dialog from the MOBUM environment.

qualitative model? The general application of all the components is discussed, as well as the basic idea and how to articulate a model.

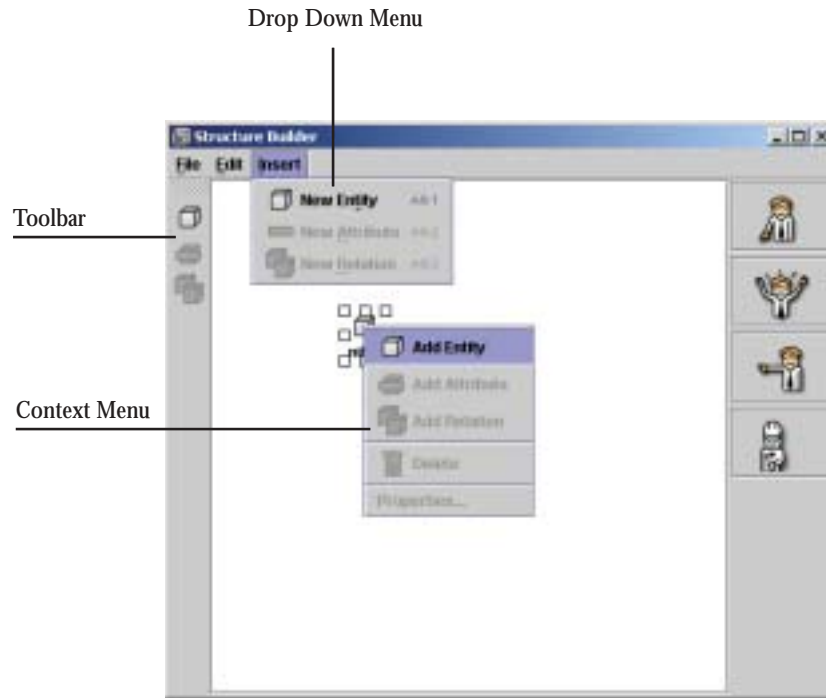
5.4.1 User Interaction

The static avatars all use the same user interaction: a dialog that displays a web page, such as figure 5.8. The avatar is situated in the upper section of the dialog. The middle section displays the web page in a window. The button for closing the dialog is placed in the bottom section. The dialog is not modal, the interaction with the remainder of the environment is not blocked, as a number of dialogs do. The modeler must be able to access the building components while obtaining information from the avatars.

The nature of the web pages enables help hyper-linking, and thus cross-referencing, through the help contents. Images connotate the text and refer to parts of the GUI.

5.5 Dynamic Support

Dynamic support provides the modeler with reflections. In contrast to the static support, the dynamic support is variable. Reflections are generated on the basis of the *contents* of the modelling environment. The dynamic support generates reflections by analysing the work of the modeler.

**Figure 5.5**

The Structure Builder in MOBUM displaying the toolbar, a context menu, and a drop down menu.

5.5.1 The Avatars

To provide the modeler with these reflections, two avatars are employed: the *what can I do next?* and the *Cross builder help...* The *what can I do next?* avatar provides local reflections, that is, local to the builder it reflects on. The *Cross builder help..* avatar provides reflections on the contents of the builder in relation to the entire model.

- *What can I do next?* This avatar reflects on the current state of the contents of the modelling component. The avatar provides intelligent support on the local aspects of the model. It actively hints on what modelling steps can be done given the current situation. The avatar is aware of the selection in the modelling component and informs the modeler about the available actions. Furthermore, the avatar suggests where to start and mentions which type of essential model constructs are missing. Figure 5.9 displays such a dialog. The modeler has just created a new entity. The avatar suggests to:

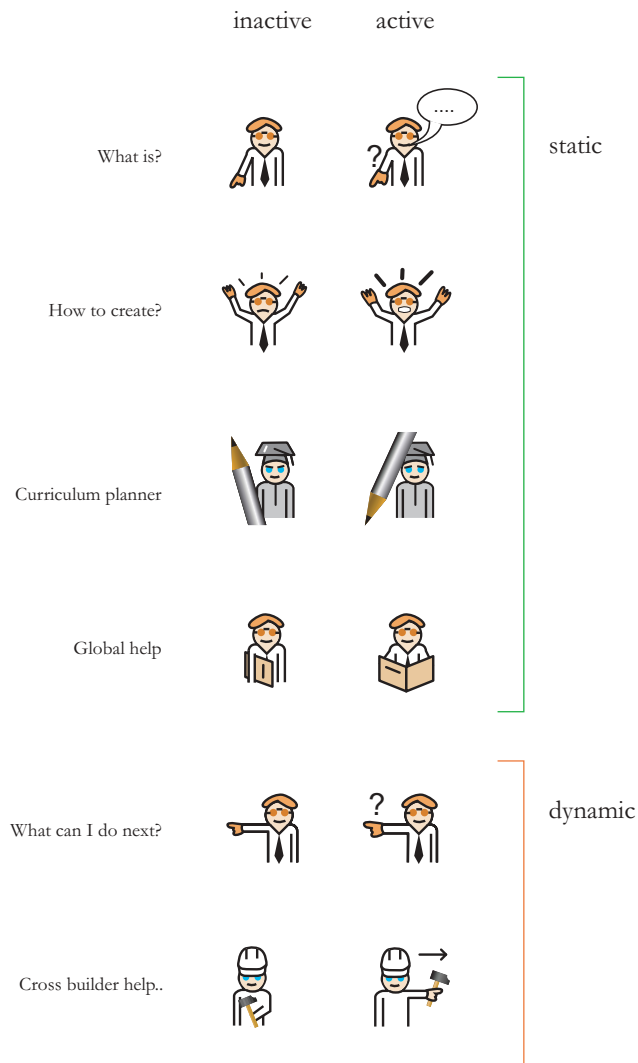
Work on the current selection You have selected an entity. The following options are possible: add a new entity as a subtype, give the entity a different name, change the super type of the entity, add an attribute, delete the selected entity, or undo your selection.

Create an attribute No attributes have been specified yet. Maybe you want to define an attribute and assign it to an entity.

Another example is the advice provided on the model fragment in figure 5.10 . The advice would be:

There are no consequences in "MF1" The model fragment "MF1" has no consequences specified, which means that no new knowledge will be added if the conditions are true. Is this what you want? Or do you want to specify some consequences that hold if the conditions are true.

Missing behavioural dependency between "B" and "A" The quantities "Q" and "P" of entities "B" and "A" are unrelated. Often quantities of different entities have behavioural relationships,

**Figure 5.6**

The six avatars acting as personified support.

specifying their dependency. Maybe you want to add a dependency between some of the quantities of these different entities.

These two rules display elaborate reflections in the model fragment builder. The first advice points out that a model fragment needs to have consequences in order to have effect. The second rule points out that it is common to have a behavioural dependency between two quantities.

- *Cross builder help..* This avatar reflects on the current state of the entire model in relation to the current modelling component. It hints on information that can be reused and notifies on unused and obsolete model parts.

For example, take the situation that the modeler has just started with a model and just created one entity. The *Cross builder help..* avatar would mention that the entity is not used in the model:

What is the Structure Model Builder?

The structure builder is used to defined entities, attributes and structural relations. After being defined in this builder these model ingredients can be re-used in other builders, particularly the model fragment and the scenario builder.

Entity

Entities represent physical objects or conceptualizations which are part of the real world problem domain. They form an important backbone to any model that is created. Entities are organized by means of a *subtype hierarchy*. The top node in the hierarchy is called *nil*. Newly added entities must be made a subtype of this nil or of other already created entities.

Examples:

- Human* is-a nil
- *Woman* is-a human
- *Physical object* is-a nil
- *Container* is-a physical object
- *Substance* is-a nil
- *Water* is-a substance

Structural relation

Structural relations model how entities are physically (or structurally) related to each other.

Examples:

- The balloon *contains* gas
- The chair *is near* the table
- The pipe *is connected to* the container

Attribute

Attributes represent static properties of entities. That is, features that usually do not change as part of the systems behavior. When modeling attributes, the set of *values* that an attribute may have must also be specified. Attributes should not be confused with quantities.

Examples:

- The *color* of a car can have values: blue, black, grey, red, etc.
- The *openness* of a container can have values: open or closed (that is: the container is open or closed, respectively).
- The *status* of a light switch can have values: on or off (and thus facilitating a current flow or not, respectively)

Figure 5.7

Text provided by the *What is?* avatar in the Structure Builder.

Unused entity "A" The entity "A" is not used in any model fragment nor in any scenario. Maybe this entity is superfluous and can be removed.

Furthermore, the *Cross builder help..* avatar is able to communicate knowledge. The modelling environment consists of specification components and construction components. In the MOBUM modelling environment, these specification components are the SWAN sketchpad and the Causal model pad. The *Cross builder help..* avatar aids the modeler by distributing the knowledge expressed in the specification components to the specific construction components. Knowledge expressed in the specification components is made reusable. The *Cross builder help..* avatar integrates the knowledge expressed in the specification components into the actual model building process.

For example, if a modeler draws a graphical object and annotates it with a name, the *Cross builder help..* avatar will refer to the object in the Structure Builder. Figure 5.11 shows a situation where the *Cross builder help..* avatar informs the user about the knowledge expressed in the SWAN sketchpad. The modeler has drawn a U-Tube and typed an object annotated as "water" as type "object". The *Cross builder help..* avatar in the Structure Builder will refer to the object "water" and explain why it

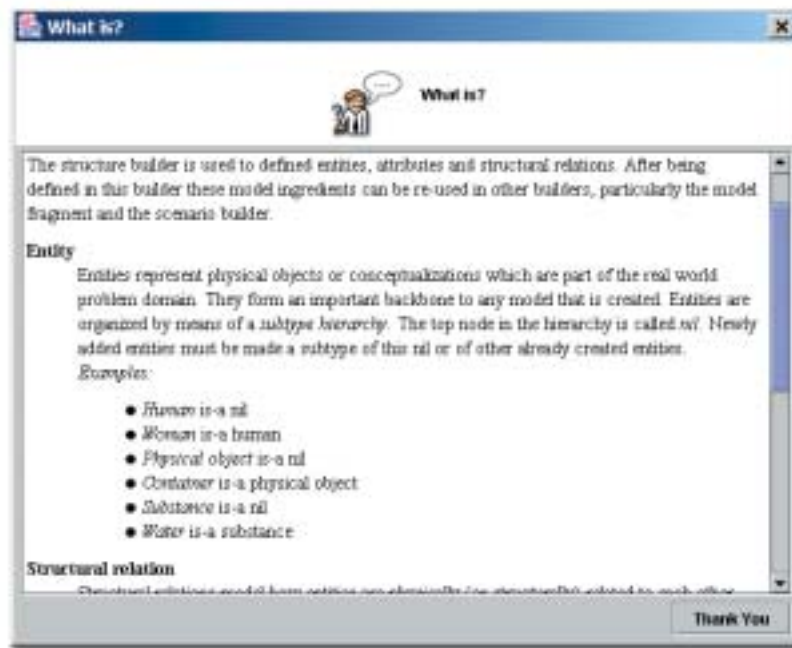


Figure 5.8

The *what is?* dialog displaying a help file..

believes it can be of interest in the structural model builder.

5.5.2 Types of Reflections

Certain reflections are more important than other reflections. Minor remarks are less important than errors. In order to provide for classification, the reflections are prioritised. We distinguish three types of reflections. Each type has a different priority. These levels of priority are: warn, advice, and inform. Note that all prior examples are of level advice.

Warn Warn the user about common errors and mistakes. The warning always attempts to present the user with an action that can be performed in order to find a solution for the problem. This type of reflection is of highest priority. An example of a warning is:

Model Fragment "MF1" is empty! The model fragment "MF2" is empty. Nothing has been specified in either the conditions nor the consequences. Maybe you want to start filling the model fragment by adding an entity as either a condition or a consequence. Or by adding an other model fragment as a condition.

Advice Advice the user on what to do next. This advice describes a certain action that can be taken. This type of reflection is of medium priority. An example of an advice in the Model Fragment Builder is:

No conditions in "MF1". The model fragment "MF1" has no conditions specified, that means that the knowledge specified as a consequence will always apply. Is this what you want? Or do you want to specify specific conditions under which these facts should be applied?

Info Inform the user on the state of the component. Info differs from advice in the sense that advice depicts user action and information does not. This type of reflection is of lowest priority. An example of an info in the Quantity Builder is:

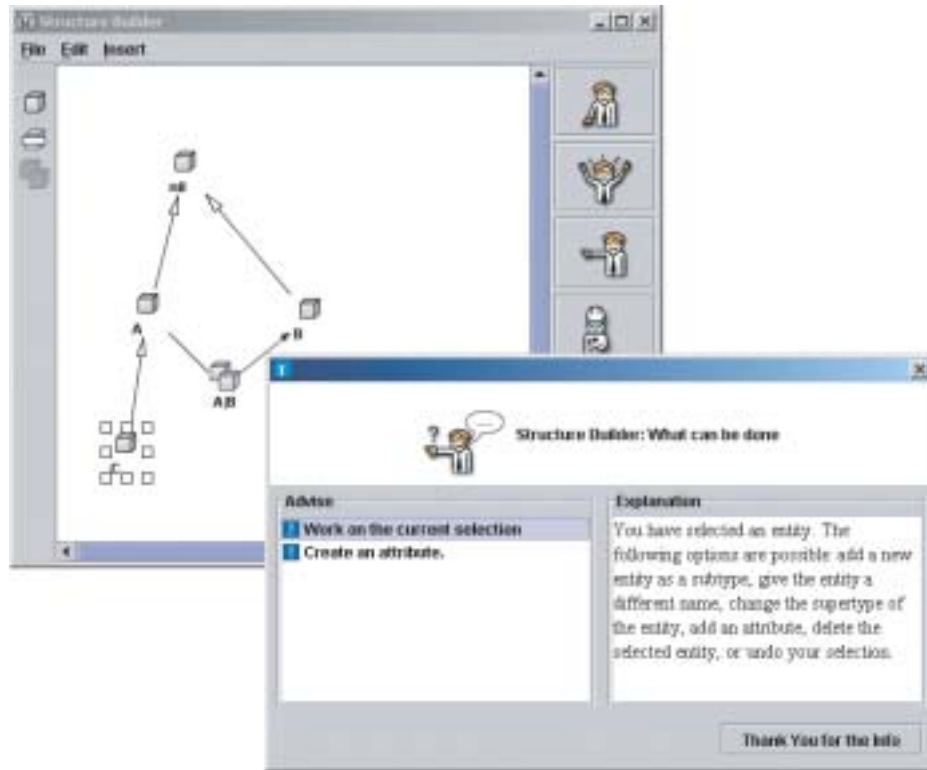


Figure 5.9

The *What can I do next?* displaying advice on the contents of the Structure Builder.

Missing quantity "A" The entity "A" has not been given a quantity. This may result in the simulation not being able to derive certain behaviour. Maybe, you want to give entity "A" a quantity.

5.5.3 User Interaction

Except for the SWAN Builder's *What can I do next?* avatar, all avatars use the same dialog. If the dialog is visible, the rule engine, described in appendix A, runs every second to update its reflections on the user actions. The dialog reacts to changes in the component within one second. The reflections are sorted by priority. Figure 5.12 displays the *What can I do next?* avatar for the Structure Builder. On the left side of the screen the titles of the advises are listed. Selecting a title displays the description of the advice.

The advice dialog is activated by the user; once it is activated, it remains visible. The dialog will not block the interaction with the rest of the environment, as certain dialogs do. The dialog adjusts to the users actions in the component. The avatar remains active until the user closes the dialog. We have deliberately chosen not to interfere in the user interaction. The avatar must be initiated by the user. We argue that the modeler should have the initiative in getting advice, the avatars should not be proactive.

The SWAN Builder local advice avatar differs from other dialogs. The SWAN local advice dialog uses a scaled image of the original sketchpad in order to graphically highlight the targets of the advice. Because a number of objects do not have a name, they need to be referred to by graphical means in order to refer to them correctly. Figure 5.13 displays the dialog giving advice on a sketch.

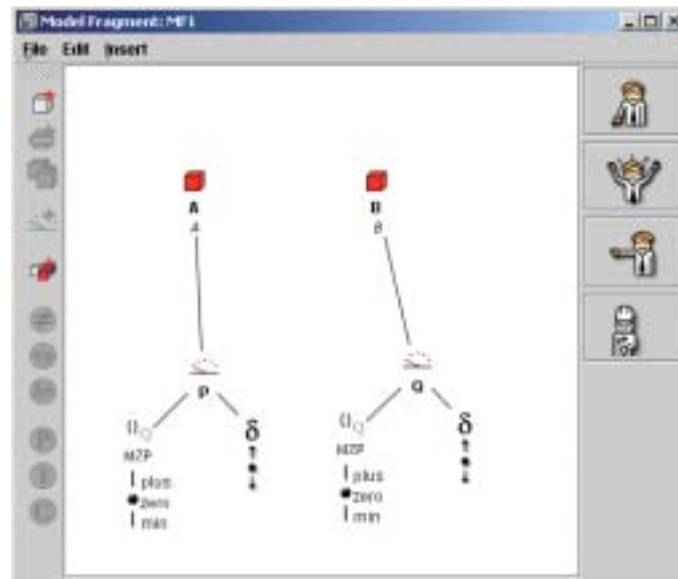


Figure 5.10
An model fragment example.

5.6 Intelligent Agents

Although the intelligent agents are discussed extensively in appendix A, the support in MOBUM cannot be discussed without mentioning the intelligent agents deployed in the modelling environment.

The dynamic avatars provide the user with reflections on the model. This reflection needs a kind of intelligent *reasoning*. Furthermore, the avatars are able to reflect on knowledge available from other components. There needs to be certain means of *communication*.

The agents framework uses one single internal representation to exchange knowledge. This *common ground of communication* facilitates the communication in the model building environment. A shared ontology is employed to make sure that all participants speak the same language. For instance, knowledge conveyed in the SWAN sketchpad is converted into the same internal representation as the knowledge in the Structure Builder is. This internal uniform representation thus facilitates an easy knowledge communication: the reasoning capabilities of the intelligent agents now have access to the knowledge needed for the inference process.

The inference engine uses rules to infer over the internal representations provided by the common ground of communication. Rules are applied to infer new knowledge. For instance, a rule specifies that if there exists an object in the SWAN sketchpad that does not occurs in the Structure Builder as an entity, there should be advice concerning this issue. The dynamic support is realised using these kind of inferences.

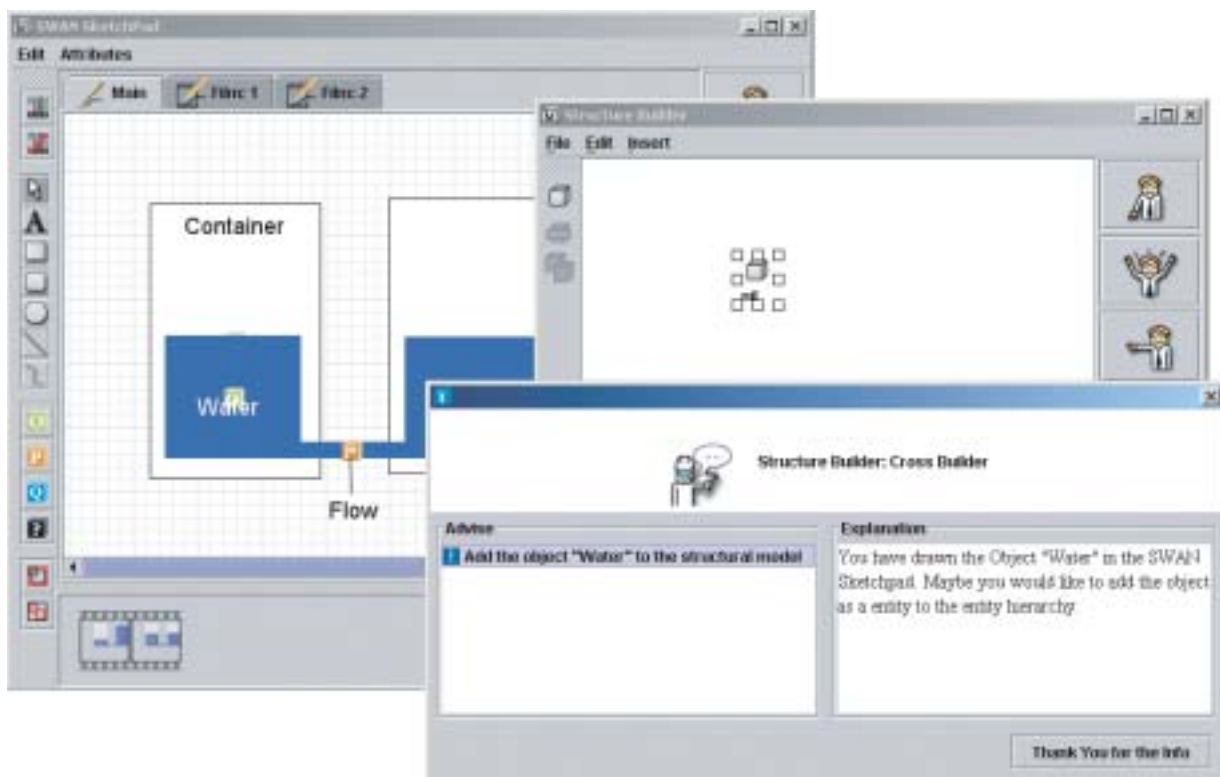
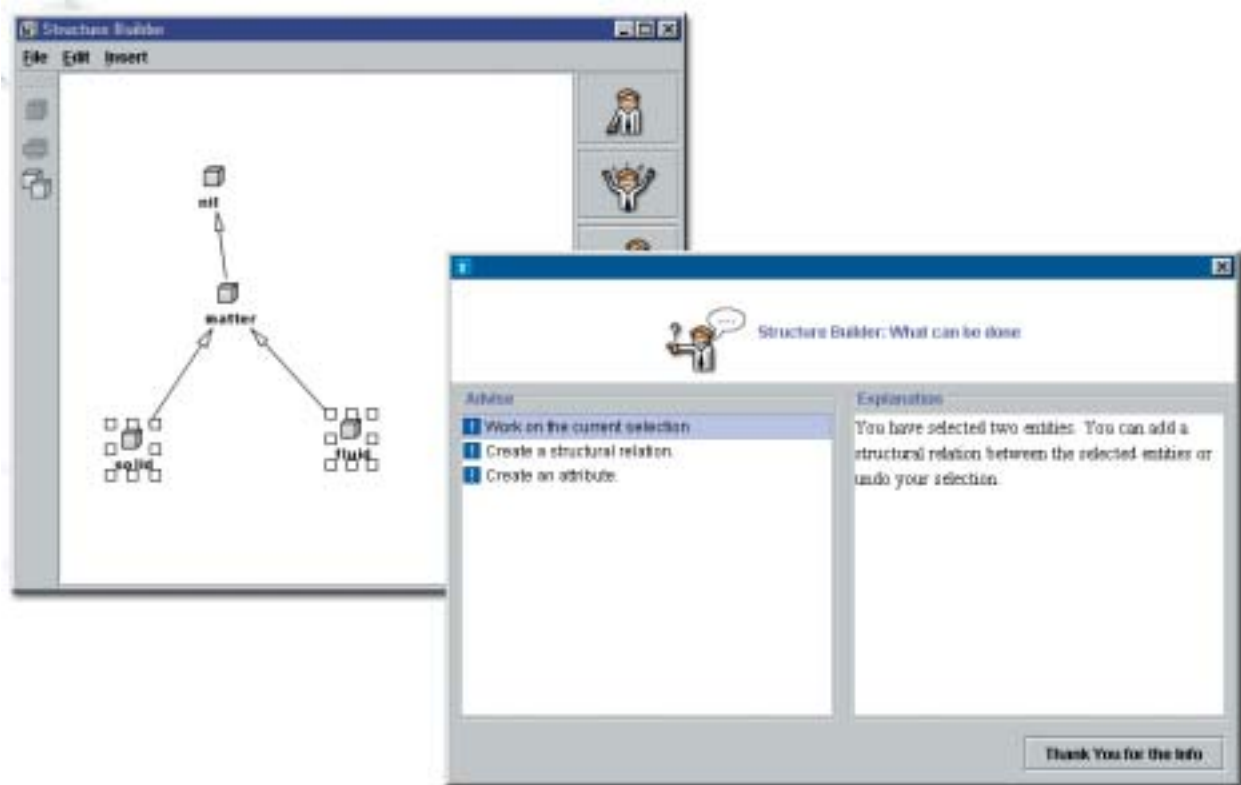


Figure 5.11

The *Cross builder help..* avatar referring to an object in the SWAN sketchpad.

**Figure 5.12**

The *What can I do next?* avatar giving reflections on the Structure Builder.

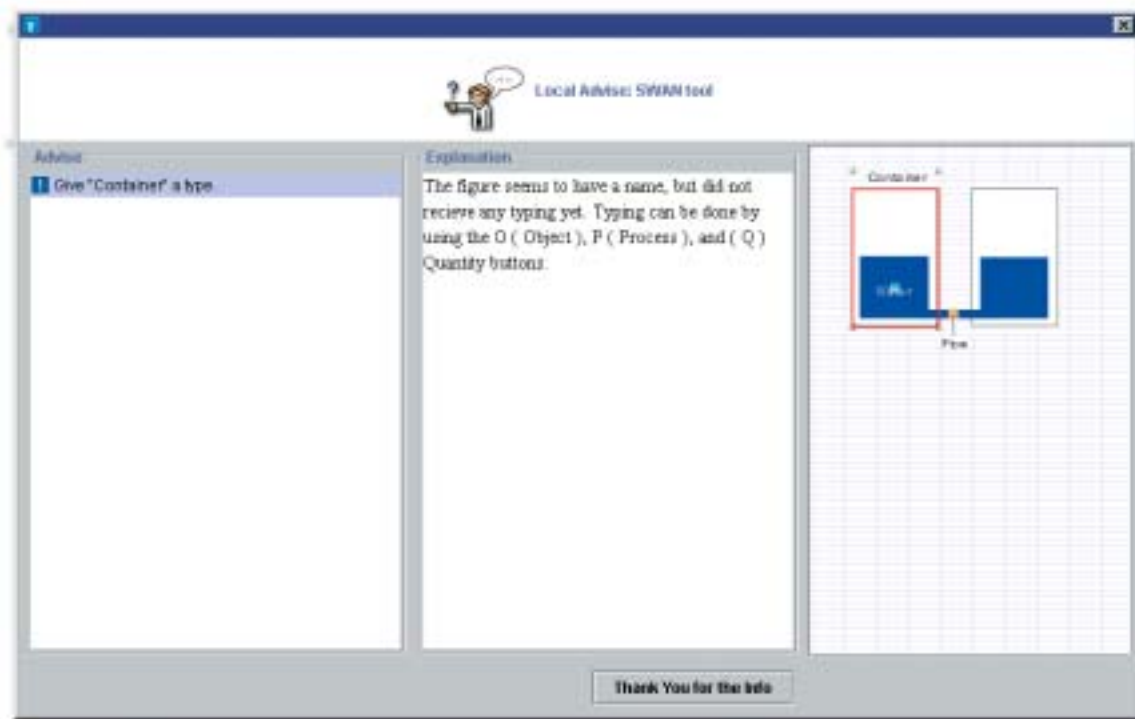


Figure 5.13

The local avatar dialog advising on a drawing.

The Experiment

6.1 The Research Question

Chapter 5 discusses how to scaffold the model building process by providing support. This chapter describes an experiment that compares two model building environments, MOBUM and HOMER. The hypothesis is that MOBUM supports learners better in their model building task than HOMER. There are two underlying ideas about the hypothesis. The first idea is that the *inherent support* has been improved. After analysing subjects with HOMER in a previous experiment [6], a number of user-interface problems have been identified which have been fixed in MOBUM. These improvements should have a positive influence on the evaluation of MOBUM. The interaction should be more fluent and intuitive and thus cause less problems and misconceptions. This is the first reason why MOBUM is expected to be evaluated more positive than HOMER. The second idea behind the hypothesis is that users like MOBUM better because it has *static* and *dynamic* support in the form of *avatars*. These avatars are able to give support for the model building task and thereby support the model builder in building models. The modeler is provided with different sorts of guidance throughout the model building task. Static avatars provide for all sorts of information such as model building basics and procedural knowledge. Dynamic avatars provide reflections on the modelers work in different ways. This should result in the modeler having less problems. It is therefore expected that MOBUM will receive a more positive evaluation than HOMER.

Furthermore, the improved support is expected to improve the model building performance. First, the inherent support makes the tool more easy to use. This will result in a more fluent and intuitive, thus quicker, user interaction. The static and dynamic support aid the model builder in the modelling task. Problems and difficulties can be resolved using the support. Model builder will have a quicker understanding of the problems at hand. This will have a positive influence on the productiveness.

6.2 The Assignment

The idea of this experiment is to test this software on non-expert, naive users, that is, unknown to the field of qualitative reasoning (QR). To give such a group the ability to use most of the functionality in a limited amount of time, the model to build was relatively simple and the assignment consisted of guiding questions. These questions started at an easy level, gradually moving into more difficult aspects of the modelling task. To give the subjects a minimal understanding of QR, a short introduction to QR is used. Subjects read this introduction in advance. The assignment is adjusted in detail to the specific tool; there is one version for HOMER and one version for MOBUM.

6.3 The Method

The experiment is conducted with 28 freshmen psychology students. There are two conditions, one starting with MOBUM, the MOBUM-HOMER (MH) condition, and one starting with HOMER, the HOMER-MOBUM (HM) condition. The two conditions differ in the order of the tools and the order in which questionnaires QM and QH are presented. The sequence of tasks and questionnaires in the two conditions of the experiment is elaborated in table 6.1.

1. The subjects are read aloud the introduction. This introduction explains the nature of the experiment, namely; a comparative evaluation of the two software tools. And furthermore, the fact that not their model building nor computer skill is tested is stressed. Their task is to evaluate the software by making the assignments and filling out the questionnaires.
2. Questionnaire A is filled in. This questionnaire is about the subjects' attitudes towards computers and automation.
3. A short English introduction to QR is read. This step takes seven minutes.
4. The subjects receive their first assignment. They use the first modelling tool for one hour to complete the assignment. For the HM group this is HOMER, for the MH group this is MOBUM. After one hour their work is saved.
5. The subjects switch to the second tool, in case of HM this is MOBUM, in case of MH this is HOMER. The subjects start over with the same assignment.
6. The subjects fill in the questionnaires. There are four questionnaires. QM evaluates MOBUM, QH evaluates HOMER. Comp compares the first tool with the second. In case of MH this questionnaire compares MOBUM with HOMER, in case of HM this questionnaire compares HOMER with MOBUM. In the MH condition the order of the questionnaires is QM and QH, in the HM condition the order of the questionnaires is QH and QM. Subsequently, questionnaires Comp and E are filled in. The Comp questionnaire compares the two tools, based on the order. In the HM condition the questionnaire compares HOMER with MOBUM, in the MH condition this questionnaire compares MOBUM with HOMER. Questionnaire E evaluates the experiment itself and asks general questions about the subject of QR and education.
7. The subjects are debriefed. The goal of the experiment is clarified and any possible questions are answered.

Condition	Questionnaire	Tasks				Questionnaires			
		7 min.	8 min.	60 min.	30 min.	15 min.			
Mobum-Homer	A	reading introduction	Mobum	Homer		QM	QH	Comp	E
Homer-Mobum	A	reading introduction	Homer	Mobum		QH	QM	Comp	E

Table 6.1

Sequence of the questionnaires and tools in the experiment.

6.4 The Means

This experiment uses two means of measurement to operationalise the research question. The first measurement is the user evaluation of the tools. This evaluation is done by questionnaires. The second measurement is the productivity measurement. This is done by assessing the created models, discussed in section 6.4.2.

6.4.1 The Questionnaires

To measure the appreciation of the tools, five questionnaires are used. One questionnaire is employed in advance of the assignment, that is questionnaire A, and four questionnaire are used after the assignment. Those are QM, QH, Comp and E.

- A** Questionnaire about the attributes towards computers, these are quite general questions about the weekly amount of the spend with computers, the attitude towards computer tasks and the preference of using the computer over watching television.

QM and QH Questionnaires for evaluating the different tools, that is, one for MOBUM and one for HOMER. The questions are similar, only the name of the tool differs. For instance, in the case of QM a question 1 looks like:

The screens in MOBUM are insightful and understandable.

1	2	3	4	5
I disagree	I somewhat disagree	neutral	I somewhat agree	I agree

And in QH the question would look like:

The screens in HOMER are insightful and understandable.

1	2	3	4	5
I disagree	I somewhat disagree	neutral	I somewhat agree	I agree

The answers that can be given by selecting five options. These options have a range from 1 to 5 starting from "I disagree" and end at "I agree". The five options are not a continuous scale, only one discrete option can be selected.

These two questionnaires consist of seven questions. To support the participant by distinguishing the two tools, pictures of the screens of the tools were provided.

Comp This questionnaire directly compares the first tool with the second tool. It differs per condition. In the HOMER-MOBUM condition HOMER is compared to MOBUM. In the MOBUM-HOMER condition MOBUM is compared to HOMER. In the first condition question 1 looks like:

The screens in HOMER are more insightful and understandable than the screens in MOBUM.

1	2	3	4	5
I disagree	I somewhat disagree	neutral	I somewhat agree	I agree

And in the second condition the question 1 would look like:

The screens in MOBUM are more insightful and understandable than the screens in HOMER.

1	2	3	4	5
I disagree	I somewhat disagree	neutral	I somewhat agree	I agree

The same answer scale as the scale of the previous questionnaires is used for answering these questions.

E This questionnaire is a general evaluation of the experiment, the assignment and the software in general. Questions like:

The questions in the assignment are formulated insightful.

1	2	3	4	5
I disagree	I somewhat disagree	neutral	I somewhat agree	I agree

This questionnaire purely has a function for gaining more understanding of the general context in which the subject placed the assignment. This will give certain insights in the subjects attitude towards the entire experiment and the level of difficulty. Most of the questions use the same five point scale.

6.4.2 The Performance Measurement

In order to measure the model building performance of the subjects, the models are scored according to a scoring system. This system is quite straightforward in enumerating model parts. These model parts are the smallest distinguishable elements in the desired model. These are at a general level entities in the hierarchy, quantity spaces, and quantities. In model fragments and scenario's: entities, quantities, quantity spaces, derivatives, and dependencies. The scoring is based on three categories of assessment:

C The correct model parts, parts complying to the assignment.

E The erroneous model parts.

S Model parts that are superfluous.

After measuring these values for each model, one in MOBUM and one in HOMER, the total score is calculated by the following formula:

$$P = C - (E + S)$$

where P is the total productivity. For each participant, the two models, one made in MOBUM and one made in HOMER, are scored using this formula of productivity measurement. If a subject has, for instance, a perfect entity hierarchy existing of three correct entities that will be scored as three correct model parts, that is 3 points. Two correct quantity spaces, and one erroneous quantity space will result in 2-1=1 points, and so on.

In spite of the straightforward model of scoring, ambiguities still exist. If, for instance, a misconception leads to multiple erroneous model parts, do we score this as one error, or as multiple errors? The first is not consistent with the scoring model, but the second will result in a *lower* score than if the subject did not even get to that part of the assignment. Eventually, our subject will have a lower score simply because he had a misconception instead of no conception at all. Certain people are more eager to try things, having a tendency to explore, and will create more (unnecessary) model parts, resulting in a lower score than people that do not have that tendency. These ambiguities make that the actual object of measurement, the performance of the students, is subject to interpretation and is not as straightforward and simple as the method is expected to be. For further discussion on model measuring and model complexity see [4].

6.5 The Prediction

6.5.1 The Evaluation

The hypothesis is that subjects evaluate MOBUM more positive in both conditions independent of the sequence of the tools they have been working with. We can further specify our prediction. There are questions in the questionnaire which are expected to be in favour of MOBUM, such as questions about help, which is more elaborated in MOBUM, and questions which are expected to be in favour of HOMER, such as questions about error messages, which are more elaborated in HOMER. Table 6.2 specifies these expectations.

nr	Question	In favor of:
1	The different screens in [Mobum Homer] are insightful and understandable in their meaning	Mobum
2	The software programm [Mobum Homer] is easy to use	Mobum
3	The use of colour in [Mobum Homer] is insightful and understandable in its meaning	both
4	The error messages in [Mobum Homer] generates are good	Homer
5	The help messages in [Mobum Homer] generates are good	Mobum
6	The use of icons in [Mobum Homer] is insightful and understandable in its meaning	Mobum
7	The navigation between the different screens in [Mobum Homer] is user-friendly	both

Table 6.2

The expectations of the results on QM and QR.

As discussed in chapter 5 the evaluation of HOMER [6] raised a number of issues concerning the graphical user interface (GUI) and user interaction. MOBUM provides more *inherent* support, discussed in section 5.2. MOBUM attempts to keep the system state visible and to make the possible actions clear to the modeler. This should result in a more easy to use modelling environment. For this reason we argue that questions 1 and 2 will be in favour of MOBUM. The use of colour is the same in both applications, so no difference between the two modelling environments is predicted. Question 3 should not be in favour of any tool. Section 5.2 discusses the different strategies to prevent the user from making errors. Although inconsistent, HOMER chooses to inform the user more often about errors. MOBUM chooses to prevent them by disabling the options that cause them. Question 4 is predicted to be in favour of HOMER because HOMER generates more and more elaborate error messages. Question 5 should be in favour of MOBUM since HOMER is not equipped with any help. Section 5.4 and 5.5 discuss how *static support* and *dynamic support* is deployed in MOBUM. Question 6 should be in favour of MOBUM. MOBUM uses icons for each builder, each basic building block and each help agent, or avatar. HOMER uses only icons for the modelling primitives. MOBUM should be favoured on question 6. Although both tools use different strategies, there is no reason to assume that one strategy is better than the other. HOMER allows for nesting dialogs and builders, that is opening a Quantity Builder to create a new quantity from the Model Fragment Builder. MOBUM does not allow this kind of nesting. MOBUM assumes the modeler knows how to find the right builder. Both strategies have advantages and disadvantages. In case of HOMER this is the fact that many nested dialogs can confuse the modeler. In case of MOBUM this problem does not occur, but the modeler must close the current dialog and find the appropriate builder to create the model part. No tool is expected to be favoured on question 7.

6.5.2 The Productivity

The inherent, static, and dynamic support in MOBUM is meant to scaffold the model builder in the model building task. An easy to use, guiding user interface combined with personified help agents, or avatars, will make learners build models more easy. For that reason we assume that learners will be able to be more productive in their task. The assumption is that they are able to do more of their assignment and create more of the desired model with less errors.

6.6 Results of the Subjects' Evaluation

This section discusses the experimental results and statistical analysis of the subjects evaluation of the tools. First, analysis of the questionnaires is discussed. Second, conclusions are drawn on these results. The complete set of tables and figures of the analysis is located in appendix B.3.

6.6.1 The Subjects Evaluation

Based on the results of questionnaires *QM* and *QH* there was a significant difference between the two conditions, the evaluation of MOBUM (mean=25, std. deviation=4.9) and evaluation of HOMER (mean=16.9, std. deviation=5.80) ($z=4.33$, $p<0.0005$), see table 6.3 and figure 6.1 for the details.

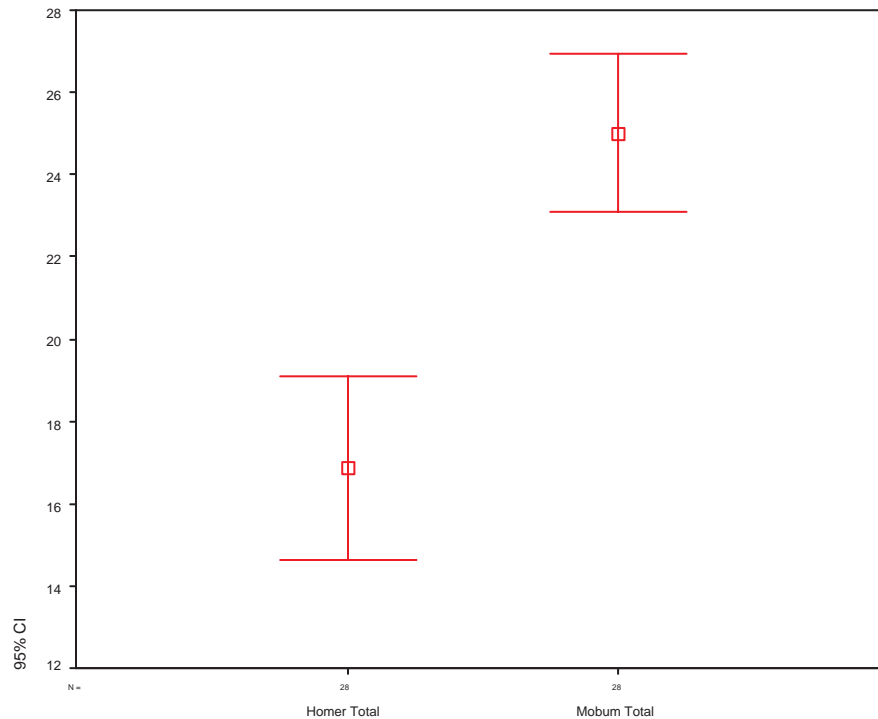


Figure 6.1

The means for the total scores on questionnaire *QM* and *QH*. The bars represent the 95% conference interval.

		N	Mean Rank	Sum of Ranks
Mobum Total - Homer Total	Negative Ranks	2(a)	4.25	8.50
	Positive Ranks	25(b)	14.78	369.50
	Ties	1(c)		
	Total	28		

	Mobum Total - Homer Total
Z	-4.339(a)
Asymp. Sig. (2-tailed)	.000

Table 6.3

The Wilcoxon test (non-parametric) on the difference between the total of HOMER and the total of MOBUM for the *QM* and *QH* questionnaires.

The reliability for the two questionnaires are sufficient (*QH* $\alpha = .831$ and *QM* $\alpha = .760$), indicating a high correlation between the items. Variance analysis tells us that the order, MOBUM first, than HOMER, or the other way around did not have any significant effect on the appreciation of any of the tools or the score in general. There is a significant preference for MOBUM over HOMER when only the first tool of

both conditions is measured ($z=2.67$, $p=0.00748$). See table 6.4 for details.

	The order of the tools	N	Mean Rank	Sum of Ranks
The first evaluated tool	Mobum-Homer	14	18.64	261.00
	Homer-Mobum	14	10.36	145.00
	Total	28		

	The first evaluated tool
Mann-Whitney U	40.000
Wilcoxon W	145.000
Z	-2.674
Asymp. Sig. (2-tailed)	.007
Exact Sig. [2*(1-tailed Sig.)]	.007(a)

Table 6.4

The Mann-Whitney (non-parametric) test for the *first* QM and QH questionnaires, that is MOBUM in the MOBUM-HOMER condition and HOMER in the HOMER-MOBUM condition.

This effect goes also for MOBUM over HOMER as only the second tool is measured ($z=3.57$, $p<0.0005$). See table 6.5 for details. The reliability of QH and QM are sufficient in their role as first tool

	The order of the tools	N	Mean Rank	Sum of Ranks
The second evaluated tool	Mobum-Homer	14	8.96	125.50
	Homer-Mobum	14	20.04	280.50
	Total	28		

	The second evaluated tool
Mann-Whitney U	20.500
Wilcoxon W	125.500
Z	-3.575
Asymp. Sig. (2-tailed)	.000
Exact Sig. [2*(1-tailed Sig.)]	.000(a)

Table 6.5

The Mann-Whitney (non-parametric) test for the *second* QM and QH questionnaires, that is MOBUM in the MOBUM-HOMER condition and HOMER .

(MOBUM $\alpha=.707$ and HOMER $\alpha=.748$) and in their role as second tool (MOBUM $\alpha=.797$ and HOMER $\alpha=.882$). Even if the tools are evaluated in their role as first or their role as second tool the effects are significant. This displays the strong tendency of the found effect.

These results are strengthened by the questionnaire *Comp*, this questionnaire compares the first tool in the condition with the second tool in the condition. The subjects are forced to make a comparison between the two tools. This questionnaire adds up to the previous two questionnaires ($z=4.41$, $p<0.0005$), but cannot be used to a single source of proof. See table 6.6 for details.

The questionnaire differs over the conditions, but is analysed as one. To do this, one must assume that the scale is symmetric, that is, that the answer 1 to a question in the MOBUM condition is equal to an answer of 5 in the HOMER condition. For example, the answer to the question:

	The order of the tools	N	Mean Rank	Sum of Ranks
Compare total	Mobum-Homer	14	21.32	298.50
	Homer-Mobum	14	7.68	107.50
	Total	28		

	Compare total
Mann-Whitney U	2.500
Wilcoxon W	107.500
Z	-4.401
Asymp. Sig. (2-tailed)	.000
Exact Sig. [2*(1-tailed Sig.)]	.000(a)

Table 6.6

The Mann-Whitney of the *Comp* questionnaire.

The screens in MOBUM are more insightful and understandable than the screens in HOMER.

1	2	3	4	5
I disagree	I somewhat disagree	neutral	I somewhat agree	I agree

can be "I agree" (5). The assumption made by this questionnaire is that this answer *is equal* to answering "I disagree" (1) to the question: HOMER condition. For example, the answer to the question:

The screens in HOMER are more insightful and understandable than the screens in MOBUM.

1	2	3	4	5
I disagree	I somewhat disagree	neutral	I somewhat agree	I agree

The dislike of a tool is viewed as equal to the liking of the other tool. And there are strong arguments against this. The two statements are not interchangeable. That is why this questionnaire is viewed as supportive to its preceding and not as an entity on its own. Although there is a methodological problem with this questionnaire, as it would be used as a single source of proof, it does add up to the proof in the light of the previous findings. These results strengthen the previous findings ($z=4.40$, $p<0.0005$). The internal reliability of the Comp list seems highly sufficient ($\alpha=.942$), indicating a strong correlation between the items.

Individual Questions

Analysis of individual questions in the MOBUM and HOMER learns that they do not behave as predicted, they are all in favour of MOBUM. And not, as predicted, several neutral and several in favour of HOMER. Of all the questions, question 4 has the least difference between the conditions, thus the tendency of the prediction is found, but not the actual behaviour.

The reliability analysis of the individual questions of QM tells us that most items correlate with the rest of the questionnaire. The reliability analysis of the questions in QH shows that most item's correlate with the rest of the test, items 5, 6, and 7 are on the low side of the spectrum, showing a mild disagreement on help, icons and navigation between the subjects.

Analysis of the reliability of the questions in Comp in figure shows one item that does not correlate highly with the rest of the items; that item is item 4. It is hard to make generalisations based on these figures, except that the items correlate highly, and the error messages show a mild disagreement between the subjects.

General Attitude Questionnaire

The first questionnaire in the experiment is about the general attitude towards computers. Table 6.7 displays the results. These findings did not have any correlation with any results or evaluation tendencies, but they can be quite informative. Most participants think they spend a normal amount of time using computers. The hours per week spent on working with the computer has its modal on 30 minutes and a mean of about 10 hours. About 70% of the subjects say they have very little programming experience, 21% said they had little and only 7% said to have some experience. There was no subject claiming to have much or very much experience with programming. 92% of the subjects say to have private access to computers, only about 8% states it does not have private access to computers. The Dutch average for 2001 is about 74%¹ having private access to computers. The majority of the subjects is mildly positive towards working with a computer. The average lays just above the middle of the scale. Furthermore, it seems that the subjects slightly prefer watching TV over working with the PC.

	N	Minimum	Maximum	Mean	Std. Deviation	Variance
Computer use	28	1.00	5.00	3.2500	1.04083	1.083
Computer hours per week	28	.50	48.00	9.7679	10.94640	119.824
Programming experience	28	1.00	3.00	1.3571	.62148	.386
Access to private computer	28	1.00	2.00	1.0714	.26227	.069
Attitude towards computer tasks	28	2.00	5.00	3.6607	.69460	.482
Computer preference over TV	28	1.00	5.00	2.5000	1.26198	1.593
Valid N (listwise)	28					

Table 6.7

Summary of the results questionnaire A.

Subjects' Evaluation of the Experiment

The final questionnaire in the experiment was a general evaluation of the experiment and the software. This questionnaire reflects on the assignment, the use of the software in general, the use of the software in educational contexts, and the use building models in educational contexts. The original data can be found in appendix B.2. Figure 6.2 displays the means of all four questions in this evaluation. Reflection on the understandability of the assignment shows a positive trend; most subjects were mildly positive on the understandability of the assignment. Only one subject thought the assignment was unclear.

Towards the question of the software promoting structured thinking about the behaviour of the system, the subjects had a mildly positive attitude.

The subjects are normally distributed in their answers. The subjects are neutral towards the thesis that building models helps understanding the behaviour of a system. The idea that interactive learning, by articulating and organising knowledge in a computer tool, is preferred over gaining knowledge by reading and listening. No subject is neutral on the question, 77% is positive to the question and 23% is negative to the question.

6.6.2 Evaluation Conclusion

It is safe to conclude that the subjects evaluated MOBUM more positive than they evaluated HOMER. This goes for the evaluation of the tools individually, questionnaire QM and QH, and as they are compared to

¹ This figure is taken from the CBS database (www.cbs.nl). The figure was taken on November 2001

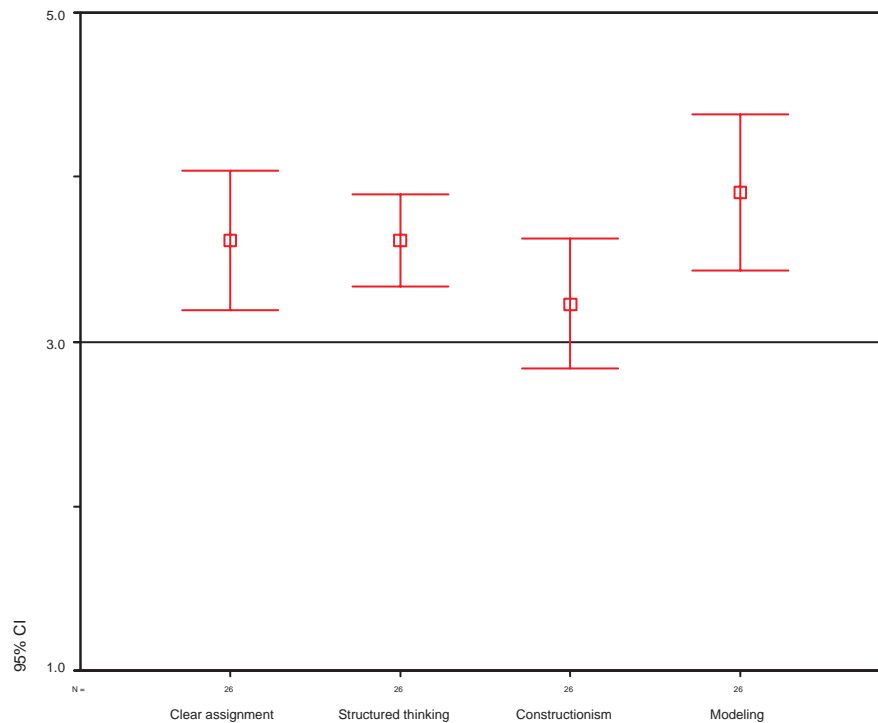


Figure 6.2

The means of all four questions of the general evaluation of the experiment. The bars represent the 95% confidence interval.

each other, questionnaire Comp. Even if only the first or the second tool is evaluated, this effect remains significant. All this together supports the hypothesis that improved usability and (intelligent) support scaffold learners in their model building tasks; subjects new to qualitative reasoning evaluate MOBUM more positively when assigned a qualitative model building task.

6.7 Results of the Subjects' Productivity

6.7.1 The Productivity Results

There seems to be no difference in measured productivity score between the models made by MOBUM and the models made by HOMER, this difference was not significant. For the correct model parts however, this difference is significant ($t=2.05$, $df=27$, $p=0.05$), as table 6.8 shows. Figure 6.3 displays the means of the correct scores of HOMER and MOBUM.

Figure 6.4 displays the same means of HOMER and MOBUM, but now ordered by condition, HM and MH. Note that subjects score better on MOBUM if it is their first tool. If they start of with HOMER, they hardly perform any better on MOBUM. The score of HOMER is low in both conditions. This suggests that there is an order effect: subjects starting of with MOBUM perform better with MOBUM than subjects starting with HOMER. However, this effect is not significant but it hints on the idea that people are somewhat confused by HOMER and are not able to recover from that. That may be the reason these subjects also preform low on MOBUM as the second tool. Subjects who start with MOBUM perform well on MOBUM, and perform normal on HOMER, as if it where their first tool.

One of the reasons for these results is that the high variance between the subjects and within the subjects, resulting in an extremely high standard deviation. The actual data can be found in appendix B.2.

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Correct scores for mobum	21.0357	28	18.08310	3.41739
	Correct scores for homer	15.1429	28	11.16779	2.11051

		N	Correlation	Sig.
Pair 1	Correct scores for mobum *	28	.545	.003
	Correct scores for homer			

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	Correct scores for mobum - Correct scores for homer	5.8929	15.21778	2.87589	-.0080	11.7937	2.049	27	.050

Table 6.8

The T-Test for comparing the *correct* of results from MOBUM with the total of results from HOMER.

Examination of this data shows an extremely high variance of the score, and thus an extremely high standard deviation, indicating why, despite of the high difference in means (MOBUM mean=18.6 st. deviation=16.6, HOMER mean=13.5 st. deviation=11.0), there is no significant effect. The created models vary highly between the individual subjects and even between the conditions of individual subjects.

6.8 Conclusion

MOBUM is more appreciated by subjects than HOMER; subjects evaluate MOBUM more positive. This goes for all the questions in the questionnaires, the a priori predicted behaviour of specific items does not seem correct. The items predicted to be neutral or in favour of HOMER, turned out to be in favour of MOBUM.

Findings are less strong when it comes to the measured performance of the subjects on the different tools. The measured performance varies high between the subjects and within the subjects, that is, between the different tools. No reliable statement can be made on the measured performance. Only when omitting the incorrect and superfluous model parts a significant effect can be found. This significant effect suggests that MOBUM is more productive than HOMER.

Based on the subjects' evaluation it is safe to conclude that improved usability and (intelligent) support influences the evaluation of the tool: subjects evaluate MOBUM more positive.

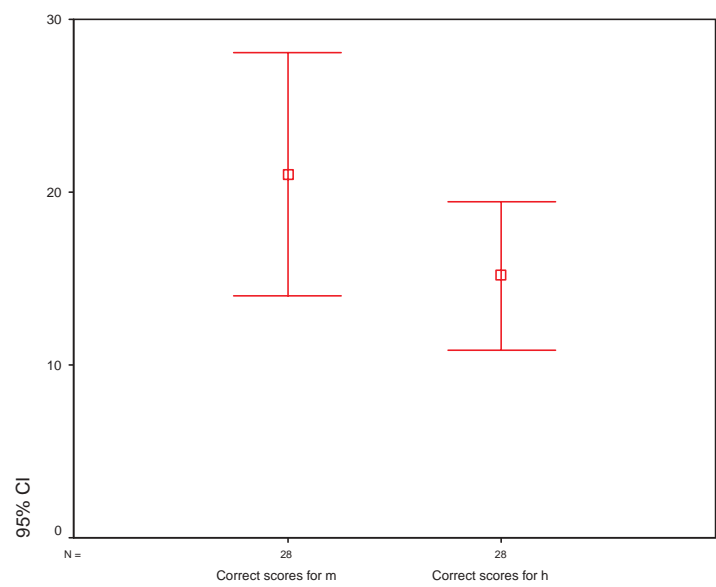


Figure 6.3
The means of the *correct* scores for HOMER and MOBUM. The bars represent the 95% confidence interval.

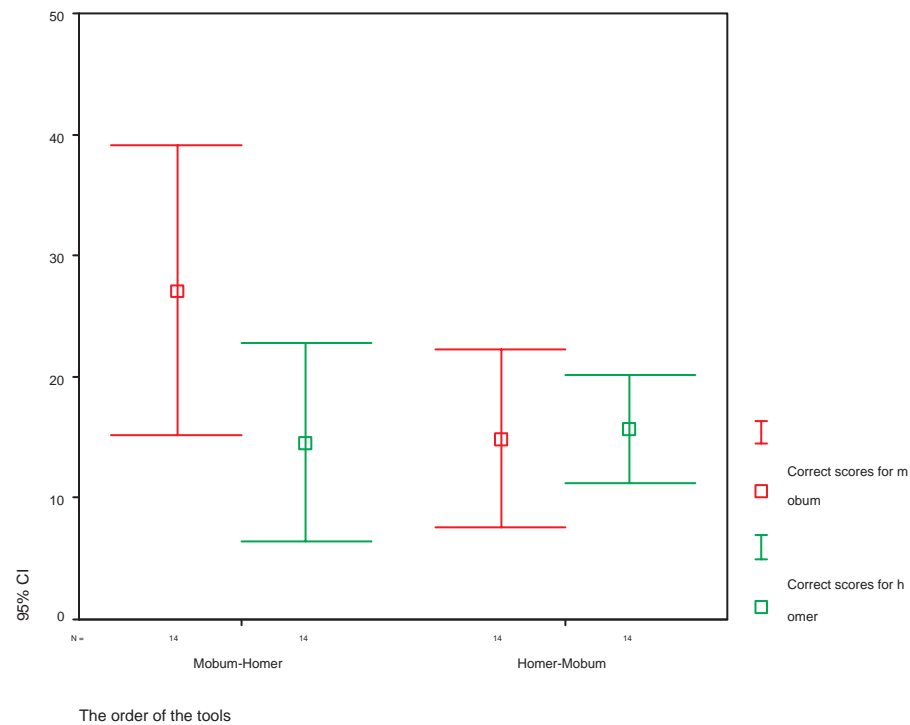


Figure 6.4
The means of the correct scores of HOMER and MOBUM by order of the tool.

Conclusion

The idea of constructionism argues that learners should learn by articulating their ideas instead of being tutored. By building models, learners acquire knowledge. The articulation of knowledge forces learners to actively express, test, reflect, and adjust their conceptions. Qualitative reasoning (QR) is a language that provides computational accounts of human reasoning and is suited for modelling in educational contexts.

In order to deploy QR in educational contexts, the software needs to be optimised for this purpose. In order to do optimise the model building environments for educational contexts, three extensions to the current model building environments are proposed: these are *specification components*, *integrated simulation inspection*, and *support*.

7.1 Specification Components

Specification components allow for model descriptions on a different level than the model building formalism itself. This thesis presents two specification components which deliver three types of specification functionality: the SWAN Sketchpad, that allows for unconstrained *sketches* and *behaviour prediction*, and the Causal Model Pad, that allows for the early creation and testing of *causal models*.

Sketching Sketching is part of the modelling process. The SWAN sketchpad is a component that provides integrated sketching. The component allows for naming and typing. The information depicted by the name and type is made available in the designated construction components, thereby integrating the sketching into the model building environment.

Behaviour Envisioning As part of the SWAN sketchpad, the behaviour envisioning allows for articulation of the envisioned behaviour of the system. A filmstrip metaphor is deployed to allow the modeler to express the envisioned states in “scenes”. Each filmstrip is a clone of its original drawing. Objects in the newly created drawing, if unchanged, remain connected with the object in the original drawing. The modeler adds, removes, or changes graphical objects to express changes with respect to the original state.

Causal Modelling The Causal Model Pad provides a simplified modelling environment, solely dedicated to expressing causal models. The Causal Model Pad only models quantities and dependencies, allowing for preliminary causal model expression. The Causal Model Pad is able to predict the behaviour of the causal model it contains and is able to present the result to the modeler. Hereby, the causal model pad allows for early feedback. The causal model pad can act as a testbed for causal models, or as a simplified modelling environment for educational purposes.

7.2 Integrated Model Prediction

In contrast to HOMER, MOBUM integrates model building with behaviour prediction and simulation inspection. MOBUM is equipped with GarpApplet, a simulation inspection tool, capable of sending a model to the GARP [14] server and visualising the contents of the simulation. The GarpApplet is part of an evolution of inspection tools dealing with the topic of complexity. The simulation can be very large, and

not all information can be presented at once. The visualisation of the state graph uses multiple methods that each visualise a specific aspect of the simulation. The visualisation of the causal model uses, besides zooming and enabling and disabling of classes of primitives, a semantic zooming method, controlled by the mouse-pointer. The visualisation becomes more detailed when hovered by the mouse pointer. This semantic fisheye aids the modeler in the task of inspecting complex simulations.

7.3 Support

Different levels of support can aid the model builder in the model building process. The system comprehends the model at hand and can scaffold the model building process by providing (intelligent) support. We distinguish three levels of support: *inherent support*, *static support*, and *dynamic support*.

Inherent support Inherent support manifests itself in how the user interface is optimised for the (modelling) task. The system state must be visible to the modeler, the available actions must be made visible, and the system's feedback plays an important role in the interaction process. Errors can be prevented by disabling options or educating the modeler by providing feedback. MOBUM implements inherent support by keeping the system state visible, making the available actions visible, and providing appropriate feedback. In comparison to HOMER, MOBUM is more consistent in error prevention and has a more optimised user interaction.

Static support Misconceptions and lack of knowledge can result in troublesome interaction. In order to provide the modeler with information that can resolve these problems, static support provides the modeler with predefined help. These help topics are available through avatars that act as personified icons. The information provided by the static help is invariable and presented in only one form.

Dynamic support Dynamic support differs from static support in its nature; the information provided is variable and generated on basis of the context. Dynamic support provides intelligent reflections on the contents of the model in progress. The dynamic support warns about problems, gives advice on common issues, and informs what options are available in the current state of the user interface. The dynamic support is accessible to the modeler through avatars.

7.4 Evaluation

The three different functional aspects of the specification components have been evaluated by modelling experts. The sketching, behaviour envisioning, and causal modelling is evaluated positive. Especially the free sketching integrated in the model building environment and the simplified causal model tool are perceived as powerful enhancements. Constructing a model is more than just articulating the model primitives; the processes surrounding and preceding this articulation of primitives should be taken in account and be supported. The specification components in MOBUM display the importance of this.

Model inspection plays an important role in the modelling *process*. By the integration of GarpApplet into MOBUM, the circle of *model* \rightarrow *test* \rightarrow *reflect* is complete. GarpApplet presents a number of advances and novel ideas in dealing with complexity in visualisations.

An experimental evaluation of HOMER and MOBUM by modelers has been conducted. The experiments shows significant results. MOBUM is evaluated more positive than HOMER. Productivity measurement suggests that MOBUM is more productive. The evaluation of MOBUM in comparison to HOMER shows that the inherent, static and dynamic support, manifested in improved user interaction and support, scaffolds the learner in the model building task and there is mild evidence that learners are more productive using MOBUM. From these results it is safe to conclude that providing support is an effective method for improving modelling environments in order for them to be effective in educational contexts.

7.5 Future Recommendations

This project does have a scope, certain aspects of the project are simply infeasible. First, The software tools presented are built as demonstration prototypes; they are not ready to be employed in real model building efforts. More resources need to be invested in order to make MOBUM end-user software. Second, the support, especially the dynamic support, is not extensive. There are uncovered topics concerning the dynamic support. One of these topics is debugging; the support is not able to reflect on the simulation.

There are some topics that future work and research on model building in educational contexts could concentrate on:

- Simplification of the modelling process, as demonstrated by the causal model pad, seems to make qualitative modelling easier to deploy in educational contexts. There are several other ideas to simplify qualitative model building. First, hiding the names of the quantity spaces from the modeler can simplify the modelling process. Naming the quantity space does not have a real function and, moreover, confuses the modeler. Second, lazy entity hierarchy creation can simplify the modelling process. The modeler can define the entities in the model fragment, and later on, if necessary, place them in a hierarchical structure. Finally, the hiding the scenario from the modeler can ease the modelling process. The scenario can be generated on basis of a model fragment. The modeler can be queried for the necessary values of the quantities.
- Modelling in automated modelling environments is highly suitable for *computer supported collaborative learning (CSCL)*. Models can be stored and communicated. (Automated) tutors can provide reflections. Furthermore, students can exchange models and even work on the same model at a distance. Intelligent agents can aid the modeler in integrating other modelers work.
- A *library of models or model parts*, such as model fragments, can aid the learner or model builder. The framework can be extended to facilitate the exchange of model elements.
- *Copying and pasting* model parts within and between construction components can improve the model building process; the modeler can reuse and duplicate configurations of primitives. This can save the modeler a considerable amount of work.

In this appendix the different aspects of the agents and the inference engine are discussed. The modelling environment consists of a collection of tools that can be divided in to construction components and specification components. Construction tools are used to build the model and often directly related to the modelling language ontology. In contrary, specification components provide abstractions to the model. These abstractions contain information that can, or should, be reused while constructing the model. Furthermore, modelers can be aided in their model building effort by providing reflections on their work. Common errors and common problems can be detected. The modeler can be made aware of these problems and can be provided with advice on how to solve them. Certain hints and tips concerning certain modelling situations can aid the modeler.

As discussed in section 2.5, agents are not agnostic, but play a role in communicating, integrating and coordinating different tasks [8]. In the modelling environment, agents provide support in two ways. First, agents help to *communicate knowledge* between different components in the environment and help to coordinate the different modelling tasks. Specification components convey abstract specifications of the model. Agents can help communicate the different types of information available in the different components. Aspects of the model that are already made explicit in a specification component can be made available in the appropriate construction component. The modeler must be able to reuse this information and be prevented from *expressing ideas twice*. The agent help coordinating the tasks in the system by facilitating knowledge exchange between tasks. Second, agents *provide intelligent reflections* on the modelers work. Agents integrate as expert advisers into the human work environment, to deliver appropriate reflections. Agents can analyse the structure of the model and provide warnings on errors, or hints on common model building issues. This refers to the active support discussed in section 5.1

In order to communicate, there must be a common ground of communication. This common ground is provided by an application program interface (API). This API can be viewed as a knowledge transport layer that provides the facilities necessary for transporting knowledge. This communication layer is discussed in section A.1

In order to provide reflections, there must be certain intelligence. This intelligence is provided by a reasoning engine. This engine is able to apply a set of rules to a knowledge structure in order to derive new knowledge, or reflections, based on the rule set. This reasoning engine is discussed in section A.2

A.1 Communication

In order to communicate knowledge there must be a common ground of communication. This common ground is provided by the Agent application program interface (API). This framework provides a formalisation of knowledge that facilitates transport and inference. The basic building blocks of the framework act as knowledge representation that is used by every agent in the system. All communicated knowledge is transformed into, or derived from these basic building blocks.

A.1.1 Building Blocks

The framework consists of knowledge objects, or *features*. These features can be instantiated and placed in a hierarchical structure, or ontology. The position in the hierarchy is determined by the *type* object.

Furthermore, these features can have *attributes*. These attributes are key-value pairs which can hold properties of the knowledge objects. Think of properties such as colour. Associations, or *predicates*, between knowledge objects are used to depict relations between the knowledge objects. A predicate connects one or more knowledge objects with a named relation. One way to think of predicates is to see them as groups of objects that belong to each other.

Predicates, features, types and attributes together form an *ontology* that facilitates for the exchange of knowledge. Since one ontology is used for exchanging knowledge, all participants in the exchange process use the same ontology in order to have a *common ground of communication* and "speak the same language".

The entire set of primitives is described as:

Features The basic knowledge object is called a *feature*. This knowledge object always has a type. This type depicts the position of the object in the ontology. A feature preferably has a name, however this is not made obligatory.

Attributes A feature can have a set of *attributes*. These are key-value pairs that can be set and read at any time.

Types A *type* depicts the position of the feature object in the ontology. The position in the ontology is based on an isa hierarchy. This hierarchy is specified at a global level, no specific hierarchies exist. There must be one root type element defined, that acts as top-level object. All elements must have a name. All except the root element must have a parent type.

Predicates Features can have associations. *Predicates* act as associations. An association can be associated with one or more features. These associations describe a kind of relationship. For instance, a group relationship can be expressed using a predicate between the grouping feature and the elements of the group.

A.1.2 Implementation

Figure A.1 displays the UML diagram of the agent API. The Feature class has a reference to exactly

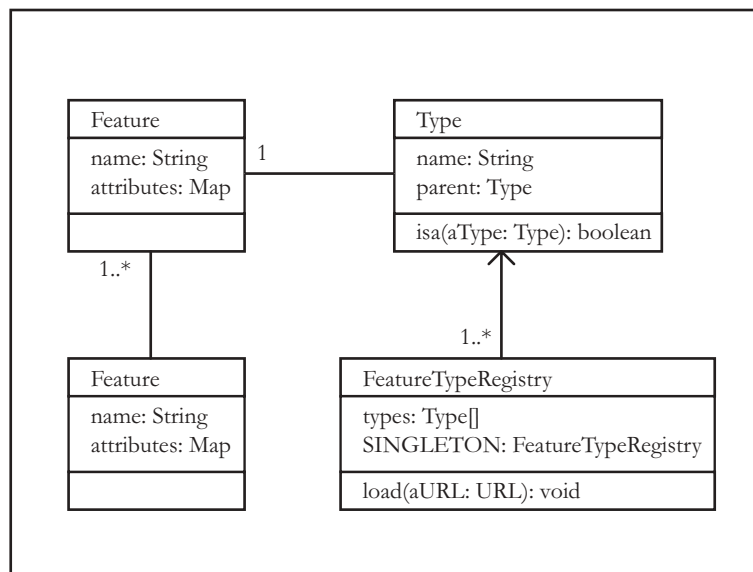


Figure A.1

The UML diagram of the basic building blocks.

one Type class instance. Types are maintained by a registry, this is the FeatureTypeRegistry. The registry takes care of the structure of the hierarchy. The hierarchy is read from a file. An example file

notation is displayed in figure A.2. The first line is obligatory: the root needs to be defined. All type's need

```
any is root;

swan isa any;

figure isa swan;

drawing isa swan;
free isa figure;
object isa figure;
process isa figure;
text isa figure;
type isa figure;
relation isa type;
group isa type;
quant isa type;
```

Figure A.2

The notation of an isa hierarchy file.

to be descendants of the root type. In the case of the example, the root types is the "any" type. The file is parsed¹ when the application initialises. The `AgentPredicate` class has at least one, or more than one, references to the `Feature` class. The `AgentPredicate` has a name which depicts the name of the predicate. The `AgentPredicate` class is not important at this moment, but it becomes crucial in the reasoning engine discussed later.

Figure A.3 depicts the translation architecture in the modelling environment. The three helper classes are responsible for the transformation of their internal knowledge representation into the knowledge representation of the Agent API. These classes perform the transformation of specific knowledge into features with types, and predicates. An example: an graphical shape classified as `Object` in the SWAN Sketchpad will be represented as a `Feature` of Type 'object' (`any.swan.figure.object`, see figure A.2). A text label overlapping the shape would be typed as 'text' (`any.swan.figure.text`). A predicate would represent the overlapping relationship. The predicate would be called 'overlapping' and would refer to both the 'text' feature and the 'object' feature.

A.2 Reasoning

In order to provide intelligent support, there must be certain reasoning capabilities. This section discusses how advice, or new knowledge, is inferred. The reasoning engine uses the knowledge representation of the Agent API. This makes the knowledge consumed and produced by the reasoning engine no different than knowledge produced by any other source that uses the Agent API knowledge representation.

The engine uses rules to infer new knowledge. The structure and workings of these rules is discussed in section A.2.1. The rules are specified in a rule language specification, discussed in section A.2.2. The specification must be compiled into an abstract representation and assembled into actual executable rules. This process of compiling is discussed in section A.2.3. Section A.2.4 discusses the internal workings of the inference engine.

A.2.1 The Rules

To derive new knowledge, the knowledge that governs the inference process needs to be described. The inference knowledge is formalised in rules. The inference engine uses the Agent API knowledge represen-

¹The parser is compiled using javacc, available at <http://www.webgain.com/products/java.cc/>.

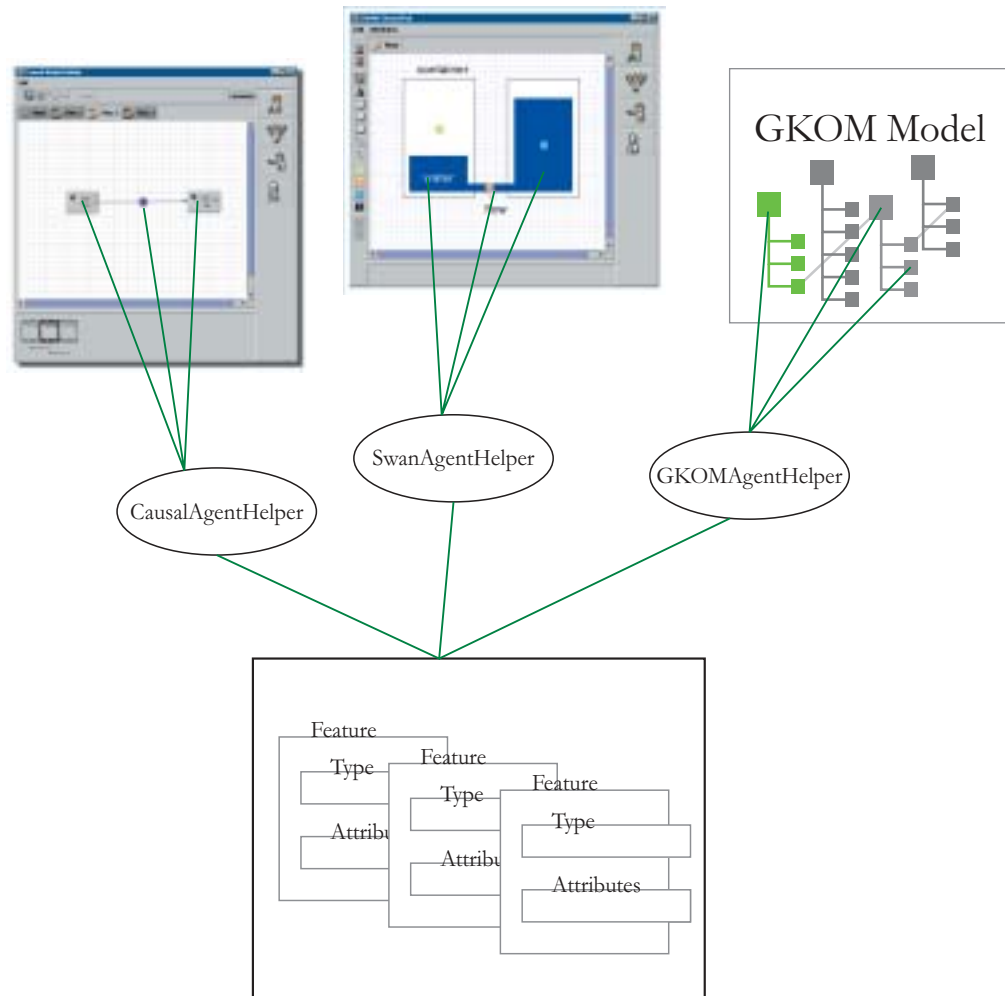


Figure A.3

The translational architecture overview.

tation. The rules use features with a certain type, features' attributes, and predicates in their description.

Rules consist of a *condition* and a *consequence*. A condition acts as the *if* part of an *if-then* statement. The consequence of a rule acts as the *then* part of an *if-then* statement. If the condition is met, the consequence becomes true.

Rules have a set of features that they apply for. This set of features, or *variables*, must be defined in the rule definition. For each feature, the variable name and type must be defined. The scope of the variable covers the entire rule definition. This is the only scope that exists: no local or global variables can be defined.

Conditions consists of one or more *statements*. A statement can be an equation between attributes of the variables, or a predicate concerning one or more variables. A statement must be true or false. A predicate statement is true if the predicate exists in the knowledge base. An equation between attributes is true when the equation holds. If more than one statement exists in the condition, they must be connected using *connection symbols*. These connection symbols are 'and' and 'or'. The 'and' symbol describes a *unification* in order for the 'and' to be true, both statements connected by the 'and' symbol must be true. The 'or' symbol describes a *disjunction* and is true if at least one of the statements it connects is true. A 'or' symbol will only result in a false statement if both statements are false. Furthermore, a statement or a group

of statements can be negated, or reversed. This results in true statements being false, and false statements being true.

Consequences consist of a predicate statement or an attribute assignment. Predicates will become true if the rule applies. Attributes will be assigned if the rule applies. Consequences can consist of many predicate statements and attribute assignments; and they do not have to be connected.

Rules do not need to have a name, instead, they can have a *title* and a *description*. The title and descriptions for certain rules are passed on to the predicates that are produced by the rules. These predicates are used to display the advice in the user interface. Names of variables can be used in the title and description. Title and description are optional.

A.2.2 The Rule Language Specification

The rules are described by a rule specification. Figure A.4 display two rules described in the rule language.

```
RuleBase {

    use object, entity_type;

    /* ID: SBC-A-support */
    Rule (object o, entity_type e) {
        if {
            o.name == e.name
        } then {
            has_entity(o);
        }
    }

    /* ID: SBC-A */
    Rule(object o) {
        if {
            o.name != "unknown" and !has_entity(o)
        } then {
            advised(o);
        } title {
            Add the object ${o.name} to the structural model
        } explanation {
            You have drawn the Object ${o.name} in the SWAN Sketchpad.
            Maybe you would like to add the object as an entity to the
            entity hierarchy.
        }
    }
}
```

Figure A.4

An Example of the rule language: two rules in an example rule base.

The Semantics: an Example

The rules use two defined types. These types are `object` and `entity_type`. These types are defined as part of the shared ontology. The first type `object` refers to graphical objects defined in the SWAN sketchpad. The second type, `entity_type` refers to entities in the Structure Builder. The first rule searches for matching graphical objects of type "object" in the swan builder with entities in the Structure Builder by their name. Each `object` with a matching entity in the structure builder will get a

`has_entity` predicate, which is added to the knowledge base. The second rule searches for instances of type `object` that do not have a `entity_type` predicate over them, and which's name attribute is not equal to "unknown". The rule searches for graphical objects in the SWAN sketchpad that do not have an entity in the Structure Builder with the same name and of which name is not "unknown". This two rules combined, inform the model builder on objects defined in the SWAN sketchpad that are not yet defined as entities in the Structure Builder.

The Syntax

A rule is defined by a `Rule` statement. The rule statement also defines the variables used in the rule. The definition of a variable consists of two parts, the type of the variable, followed by the variable name. The name must be unique within the rule. Multiple variable definitions are separated by a comma.

The obligatory structure of a rule consist of a `if` (conditional) block, containing a statement, and an `then` (consequence) block. A `then` block does not need to have any content, but a rule would be senseless without it. A `then` block can attribute assignments and predicate statements.

The first rule in the example has two variables, a feature of type 'object' named 'o' in this rule, and a feature of type 'entity_type' named e. These two variables are accessible through the entire rule. The condition of the first rule is an attribute equation. The equation states that the name of 'o' and the name of 'e' must be equal. If this condition is found to be true, the condition will apply. In case of this rule, the consequence is a predicate statement. The predicate is named 'has_entity' and applies to 'o'. Thus, if the rule applies, the predicate 'has_entity' will be applied to the Feature 'o'. If multiple consequence statements exist, they must be separated by a semicolon. This rule will add the predicate `has_entity` to every feature of type 'object' if one equally named feature of type 'entity_type' exists. This rule is used to detect if a graphical shape of type `object` already has an entity in the model with the same name.

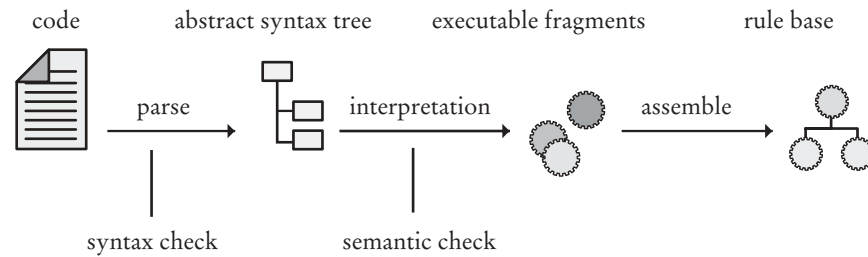
The second rule in the example has only one variable, namely 'o' of type 'object'. The condition consists of a connection symbol. An 'and' symbol connects two statements. The statement on the left-hand-side states that the name of the object 'o' cannot be "unknown". Note that attributes are accessed by a `<variable>.<attribute>` notation. The right-hand-side statement states that there cannot be a predicate 'has_entity' that applies over 'o'. A statement can be negated by an exclamation mark (!). If this condition is met, the consequence adds an 'advised' predicate over 'o'. In contrast to the first rule, this rule does have a title and description. Note that a parameter from a variable is inserted into the title by the `${<variable>.<attribute>}` notation.

Rules are grouped into one `RuleBase`. Typically, rule bases are put in separate files. A rule base consists of one or more rules. The rule base file furthermore consists of a statement that describes what types are used in the rule base. This line starts with the `use` element, the types are separated by a comma and the statement is terminated by a semicolon. If the types used in the rules are not defined in the `use` statement, the rule base is not correct. This forces the programmer to be explicit about the types that are being used in the rule base.

A.2.3 The Compiler

The rule base compilation process consists of three steps: parsing, translation, and assembly. Figure A.5 displays the compilation process. First, as source code file is read and *parsed* into *tokens*. Tokens are the language specification primitives. Each token is of a certain type. The source code is scanned for tokens and these tokens are mapped into an hierarchy, depending on the type and location of the token in the source file. The hierarchy of tokens is called the *abstract syntax tree*. The parser uses the language specification to discover the tokens in the source code. Subsequently, the parser checks the *syntax* of the file. If the structure of the file is not correct, the parser fails to determine the tokens in the source file and fails to parse the source file. If the parser succeeds parsing the source file, the syntax of the file meets with the rule language specification.

Second, the abstract representation is *interpreted* into executable elements. This process translates the tokens in the abstract syntax tree into a *executable representation*. At this stage, the executable rule objects are created in memory, and the semantical structure becomes apparent. During this interpretation process, a

**Figure A.5**

The parsing, translating, and assembling process of the rule base compiler..

semantical check on the structure is performed. Type checking on the variables is performed. For example, if a rule uses a unknown variable type, the parsing process will generate an error and terminate.

Finally, the executable representation is *assembled* into a rule base instance. The rules are placed in the right structure and the rule base is prepared for execution. At this stage there is no additional checking.

A.2.4 Inference Engine

The inference engine takes a rule base and a knowledge base as input. A rule base represents the procedural knowledge needed for the engine to infer. A knowledge base represents the facts, or knowledge, available to the inference process.

Rule Base As described in the previous section, a rule base is compiled from source code file. This rule base consist of the rules that are used to infer new facts into the knowledge base. The RuleBase class facilitates the reasoning engine in search and retrieval methods, it also keeps track of the state of applied rules in the reasoning progression. A rule base acts as a rule repository. For each distinct context a different rule base, or combination of rule bases, can be deployed.

Knowledge Base The knowledge base consists of instances of the Feature class and Predicate class. The features are placed in an isa-hierarchy by a instance of a Type class. Features can have attributes. Predicates convey a relation between one or more Features. Predicates have a unique name by which they can be addressed. The knowledge base, consisting of features and predicates, acts as a storage for facts. Each feature can be viewed as an object and each predicate can be viewed as a relation about one or more objects. For instance, the predicate ‘father_of’, conveys a kind of relationship over the features of type person Jan and Kees. The inference engine uses predicates and features for the rule base to infer upon. New inferred facts are in turn added to the knowledge base. The KnowledgeBase class is optimised for search and retrieval methods to facilitate the reasoning engine in its reasoning process.

Figure A.6 depicts how the process evolves in the entire architecture. The inference starts off with an initial knowledge base. This initial knowledge base is derived from the model, and components in the modelling environment . The first reasoning step is searching for rules that apply.

That process takes a knowledge base and a rule base as input, and finds out what rules apply. These rules are *executed*. This execution will add new knowledge to the knowledge base. This new knowledge could be new predicates or new attribute values. The next step will be searching for rules that apply in the changed knowledge base. There is a record of the rules that have been applied, in order to prevent from applying the same rules twice. The process will continue searching and applying for rules, until there are no more rules that apply.

As figure A.4 depicts, advices are actually predicates of a certain class. At the end of the reasoning process, the knowledge base is queried for predicates of that class. The producing rule provides them with a description and a title. These title and description are used in the graphical user interface (GUI).

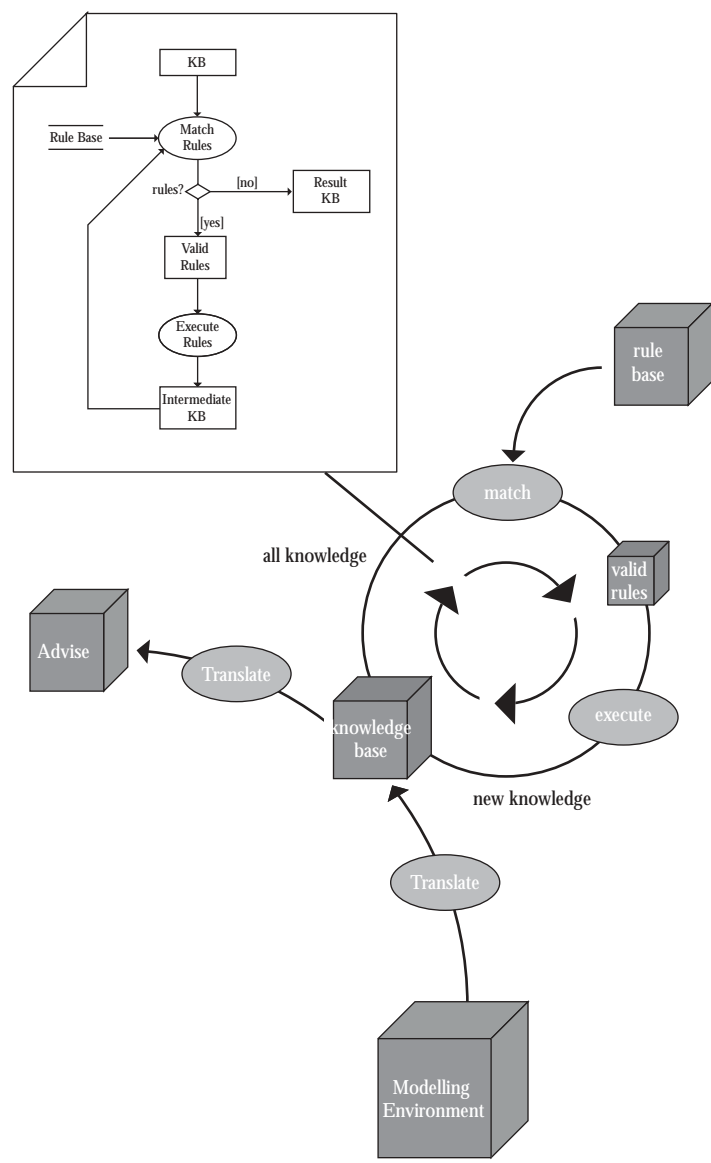


Figure A.6
Reasoning Engine overview. The note on the left uses the CommonKADS notation, as used in [55].

A.3 The Complete Rule Base

A.3.1 The SWAN Sketchpad

The SWAN Pre Rule Base

```
RuleBase {

    use any, group, text, quant, drawing, figure;

    /*
     * If a figure in group with text, the value of the text =
     * the name of your figure..
     */
    Rule (figure f, text t, group g) {
        if {
            in_group(g, f) and in_group(g, t) and f != t
        } then {
            f.name = t.text;
            named(f);
        }
    }

    /*
     * If a figure and text are connected by another figure,
     * the value of the text = the name of your figure..
     */
    Rule (figure a, figure b, text t) {
        if {
            intersecting(a, b)
            and intersecting(b, t) and t != a
        } then {
            a.name = t.text;
            named(a);
        }
    }

    /*
     * If a figure intersects with text, the value of the
     * text = the name of your figure..
     */
    Rule (figure a, text t) {
        if {
            intersecting(a, t) and t != a
        } then {
            a.name = t.text;
            named(a);
        }
    }

    /*
     * If a figure overlaps a text, the value of the
     * text = the name of your figure..
     */
    Rule (figure a, text t) {
        if {
            contains(t, a)
        } then {
            a.name = t.text;
            named(a);
        }
    }

    /*
     * If no name rule applies: delete the name....
     */
    Rule (figure a) {
        if {
            !named(a);
        } then {
            a.name = a.bull;
        }
    }
}
```

The SWAN Local Rule Base

```
RuleBase {

    use any, group, text, quant, drawing, figure;

    Rule (quant f, drawing d) {
        if {
            in_drawing(f, d) and current(d)
            and f.name=="unknown"
        } then {
            advised(f);
        }
        title {
            Give this quantity ${f.name} a name.
        }
        explanation {
            The quantity does not seem to have a name, to give a
            name, make a group with a text figure.
        }
    }
}
```

```
Rule (object f, drawing d) {
    if {
        in_drawing(f, d) and current(d)
        and f.name=="unknown"
    } then {
        advised(f);
    }
    title {
        Give this object a name.
    }
    explanation {
        The figure does not seem to have a
        name, to give a name.
        a. make a group with a text figure,
        b. put a text figure inside the figure
        c. make a connecting line with a text figure.
    }
}

Rule (drawing d) {
    if {
        !has_figures(d) and current(d)
    } then {
        advised(d);
    }
    title {
        Be creative!
    }
    explanation {
        The drawing does not contain anything,
        add something by using the drawing tools.
    }
}

Rule (typeable f, drawing d) {
    if {
        in_drawing(f, d) and current(d) and
        f.name!="unknown"
    } then {
        advised(f);
    }
    title {
        Give ${f.name} a type.
    }
    explanation {
        The figure seems to have a name, but did not receive
        any typing yet. Typing can be done by using the O
        (Object), P (Process), and (Q) Quantity buttons.
    }
}
}
```

A.3.2 The Causal model pad

The Causal Cross Rule Base

```
RuleBase {

    use any, group, text, quantity;

    /**
     * If a text and a quantity are in the same group,
     * set the text property of the text element into
     * the name property of the quantity element.
     */
    Rule (group g, text t, quantity q) {
        if {
            of_group(g, t) and of_group(g, q)
        }
        then {
            q.name = t.text;
            //in_group(g, a);
        }
    }

    Rule (quantity q) {
        if {
            in_swan(q) and (!in_causal(q))
            //in_swan(q)
        } then {
            advised(q);
        }
        title {
            Add the quantity ${q.name} to the causal model.
        }
        explanation {
            You seem to have defined a quantity in your drawing,
            you might want to use that quantity to model some
            causality between them.
        }
    }
}
```

A.3.3 The Structure Builder

The Structure Local Rule Base

```
RuleBase {

    use any, entity_type, model, attribute_type;
```

```

/**
 * local
 * id: SBL1
 * version: 1.0
 * rule content (fact):
 * If the structural builder does not have any entities
 * feedback (fact):
 * Create an entity explanation
 * The structural builder is empty. Nothing has been
 * specified yet. You can start by defining an entity.
 */
Rule(model m) {
  if {
    !has_entities(m)
  } then {
    warn(m);
  } title {
    Start by specifying an Entity.
  } explanation {
    The structural builder is empty. Nothing has been
    specified yet. You can start by defining an entity.
  }
}

/**
 * local
 * id: SBL2
 * version: 1.0
 * rule content (heuristic):
 * If the structural builder does not have any attributes,
 * and at least one entity exists.
 * feedback (suggestion):
 * Create an attribute
 * explanation
 * No attributes have been specified yet. Maybe you want
 * define an attribute and assign it to an entity.
 */
Rule(model m) {
  if {
    has_entities(m) and
    !has_properties(m)
  } then {
    advised(m);
  } title {
    Create an attribute.
  } explanation {
    No attributes have been specified yet. Maybe you
    want define an attribute and assign it to an entity.
  }
}

/**
 * local
 * id: SBL3
 * version: 1.0
 * rule content (heuristic):
 * If the structural builder does not have any structural
 * relations, and two (or more) entities exist (E1 & E2), and
 * E1 and E2 do not have a subtype relation with each other.
 * feedback (suggestion):
 * Create a structural relation
 * explanation
 * No structural relations have been specified yet. Maybe you
 * want define a structural relations between two entities.
 */
Rule(model m, entity_type e1, entity_type e2) {
  if {
    !has_relations(m)
    and !subtype_of(e1, e2)
    and !subtype_of(e2, e1)
    and e1 != e2
  } then {
    advised(m);
  } title {
    Create a structural relation.
  } explanation {
    No structural relations have been specified yet.
    Maybe you want define a structural relations between
    two entities.
  }
}

/**
 * local
 * id: SBL4
 * version: 1.0
 * rule content (heuristic):
 * If an entity is selected
 * feedback (suggestion):
 * Work on the current selection
 * explanation
 * You have selected an entity. The following options are
 * possible: add a new entity as a subtype, give the entity a
 * different name, change the supertype of the entity, add an
 * attribute, delete the selected entity, or undo your
 * selection.
 */
Rule(drawing d, entity_type e) {
  if {
    d.selection=="1"
    and selected(e) and e.name!="nil"
  } then {
    advised(d);
  }
}

} title {
  Work on the current selection
} explanation {
  You have selected an entity. The following options
  are possible: add a new entity as a subtype, give
  the entity a different name, change the supertype of
  the entity, add an attribute, delete the selected
  entity, or undo your selection.
}
}

/**
 * local
 * id: SBL5
 * version: 1.0
 * rule content (heuristic):
 * If an attribute is selected
 * feedback (suggestion):
 * Work on the current selection
 * explanation
 * You have selected an attribute. The following options are
 * possible: give the attribute a different name, change
 * attribute values, delete the selected attribute, or undo
 * your selection.
 */
Rule(drawing d, attribute_type a) {
  if {
    d.selection=="1"
    and selected(a)
  } then {
    advised(d);
  } title {
    Work on the current selection
  } explanation {
    You have selected an attribute. The following
    options are possible: give the attribute a different
    name, change attribute values, delete the selected
    attribute, or undo your selection.
  }
}

/**
 * local
 * id: SBL6
 * version: 1.0
 * rule content (heuristic):
 * If a structural relation is selected
 * feedback (suggestion):
 * Work on the current selection
 * explanation
 * You have selected a structural relation. The following
 * options are possible: give the relation a different name,
 * change the entities related by the relation, delete the
 * selected structural relation, or undo your selection.
 */
Rule(drawing d, structural_relation r) {
  if {
    d.selection=="1"
    and selected(r)
  } then {
    advised(d);
  } title {
    Work on the current selection
  } explanation {
    You have selected a structural relation. The
    following options are possible: give the relation a
    different name, change the entities related by the
    relation, delete the selected structural relation,
    or undo your selection.
  }
}

/**
 * local
 * id: SBL7
 * version: 1.0
 * rule content (suggestion):
 * If two entities are selected
 * feedback (suggestion):
 * Work on the current selection
 * explanation
 * You have selected two entities. You can add a structural
 * relation between the selected entities or undo your
 * selection.
 */
Rule(drawing d, entity_type e1, entity_type e2) {
  if {
    d.selection=="2"
    and selected(e1)
    and selected(e2)
    and e1 != e2
  } then {
    advised(d);
  } title {
    Work on the current selection
  } explanation {
    You have selected two entities. You can add a
    structural relation between the selected entities or
    undo your selection.
  }
}
}

```

The Structure Cross Rule Base

```

RuleBase {

    use any, entity_type, model, attribute_type, object;

    /*
     * If a figure overlaps a text, the value of the
     * text = the name of your figure..
     */
    Rule (object o, entity_type e) {
        if {
            o.name == e.name
        } then {
            has_entity(o);
        }
    }

    /* ID: SBC-A */
    Rule(object o) {

        if {
            o.name != "unknown" and !has_entity(o)
        } then {
            advised(o);
        } title {
            Add the object ${o.name} to the structural model
        } explanation {
            You have drawn the Object ${o.name} in the SWAN
            Sketchpad. Maybe you would like to add the object as
            an entity to the entity hierarchy.
        }
    }

    /**
     * global: wrt model Fragment Builder and Scenario Builder
     * id: SBG1
     * version: 1.0
     * rule content (heuristic):
     * For each entity E that exists in the structural builder and
     * that is not used in any model fragment and not used in any
     * scenario:
     * feedback (suggestion):
     * Unused entity
     * explanation
     * The entity E is not used in any model fragment nor in any
     * scenario. Maybe this entity is superfluous and can be
     * removed.
     */
    Rule(entity_type e) {
        if {
            !in_mf(e)
            and !in_scenario(e)
            and !e.name=="nil"
        } then {
            advised(e);
        } title {
            Unused entity ${e.name}
        } explanation {
            The entity ${e.name} is not used in any model
            fragment nor in any scenario. Maybe this entity is
            superfluous and can be removed.
        }
    }

    /**
     * global: wrt model Fragment Builder and Scenario Builder
     * id: SBG2
     * version: 1.0
     * rule content (heuristic):
     * For each attribute A that exists in the structural builder
     * and that is not used in any model fragment and not used in
     * any scenario:
     * feedback (suggestion):
     * Unused attribute
     * explanation
     * The attribute A is not used in any model fragment nor in
     * any scenario. Maybe this attribute is superfluous and can
     * be removed.
     */
    Rule(attribute_type a) {
        if {
            !in_mf(a)
            and !in_scenario(a)
            and !a.name=="nil"
        } then {
            advised(a);
        } title {
            Unused attribute ${a.name}
        } explanation {
            The attribute ${a.name} is not used in any model
            fragment nor in any scenario. Maybe this attribute
            is superfluous and can be removed.
        }
    }

    /**
     * global: wrt model Fragment Builder
     * and Scenario Builder
     * id: SBG3
     * version: 1.0
     * rule content (heuristic):

```

```

     * For each structural relation SR that exists in the
     * structural builder and that is not used in any model
     * fragment and not used in any scenario:
     * feedback (suggestion):
     * Unused structural relation
     * explanation
     * The structural relation SR is not used in any model
     * fragment nor in any scenario. Maybe this structural
     * relation is superfluous and can be removed.
     */
    Rule(structural_relation_type r) {
        if {
            !in_mf(r)
            and !in_scenario(r)
            and !r.name=="nil"
        } then {
            advised(r);
        } title {
            Unused structural relation ${r.name}
        } explanation {
            The structural relation SR is not used in any model
            fragment nor in any scenario. Maybe this structural
            relation is superfluous and can be removed.
        }
    }
}

```

A.3.4 The Quantity Builder

The Quantity Local Rule Base

```

RuleBase {

    use any, entity, model, attribute, model_fragment, quantity;

    /**
     * local
     * id: QBL1
     * version: 1.0
     * rule content (fact):
     * If no quantity exist
     * feedback (fact):
     * Create a quantity
     * explanation
     * The quantity builder is empty. No quantities have been
     * specified yet. You can start by defining a quantity.
     */
    Rule(model m) {
        if {
            !has_quantities(m)
        } then {
            warn(m);
        } title {
            Create a quantity.
        } explanation {
            The quantity builder is empty. No quantities have
            been specified yet. You can start by defining a
            quantity.
        }
    }

    /**
     * local
     * id: QBL2
     * version: 1.0
     * rule content (heuristic):
     * If one or more quantities exist (NB: assuming that in the
     * quantity builder always one quantity is selected)
     * feedback (suggestion):
     * Create or modify a quantity
     * explanation
     * You have selected a quantity. The following options are
     * possible: give the quantity a different name, give the
     * quantity a different quantity space, delete the quantity,
     * select an other quantity, or create a new quantity.
     */
    Rule(model m) {
        if {
            has_quantities(m)
        } then {
            advised(m);
        } title {
            Create or modify a quantity
        } explanation {
            You have selected a quantity. The following options
            are possible: give the quantity a different name,
            give the quantity a different quantity space, delete
            the quantity, select an other quantity, or create a
            new quantity.
        }
    }

    /**
     * local
     * id: QBL3
     * version: 1.0
     * rule content (fact):
     * If only one quantity exist

```

```

    * feedback (fact):
    * Create a second quantity
    * explanation
    * Only quantity has been defined in the quantity builder.
    * Although correct, many simulations require two or more
    * quantities. Maybe you want to define a second quantity.
    */
Rule(model m) {
    if {
        has_quantities(m)
        and m.quantities=="1"
    } then {
        advised(m);
    } title {
        Create a second quantity
    } explanation {
        Only quantity has been defined in the quantity
        builder. Although correct, many simulations require
        two or more quantities. Maybe you want to define a
        second quantity.
    }
}
}

```

The Quantity Cross Rule Base

```

RuleBase {
    use any, entity, model, attribute, model_fragment, object, quant;

    /**
     * If a figure overlaps a text, the value of the
     * text = the name of your
     * figure..
     */
    Rule (quant o, quantity_type q) {
        if {
            o.name == q.name
        } then {
            has_quantity(o);
        }
    }

    /**
     * global: wrt SWAN Builder
     * id: QBGL-SWAN
     * version: 1.0
     * rule content (heuristic):
     * For each quantity Q that exists in the SWAN builder and
     * that has not been specified in the quantity builder (NB:
     * assume 'name' matching although this may not be sufficient.
     * E.g. if a model builder uses 'pressure left',
     * 'pressureleft', 'pressure right', etc. we probably only
     * want to add only 'pressure' as a quantity type. However, if
     * we use name-matching all these quantities will be found by
     * this rule.):
     * feedback (suggestion):
     * Create a quantity
     * explanation
     * The quantity Q is used in the causal model builder. Maybe
     * this quantity should be defined as a quantity type in the
     * quantity builder (notice, that only quantity types should
     * be defined in the quantity builder and not specific
     * versions of that quantity type).
     */
    Rule(quant o) {
        if {
            o.name != "unknown"
            and
            !has_quantity(o)
        } then {
            advised(o);
        } title {
            Add the quantity ${o.name} as a Quantity to the model
        } explanation {
            You have drawn a figure called ${o.name} in the
            SWAN SketchPad and typed it as a Quantity. You might
            want to add this figure.
        }
    }

    /**
     * global: wrt Causal Model Builder
     * id: QBGL-CM
     * version: 1.0
     * rule content (heuristic):
     * For each quantity Q that exists in the causal model builder
     * and that has not been specified in the quantity builder
     * (NB: assume 'name' matching although this may not be
     * sufficient. E.g. if a model builder uses 'pressure left',
     * 'pressureleft', 'pressure right', etc. we probably only
     * want to add only 'pressure' as a quantity type. However, if
     * we use name-matching all these quantities will be found by
     * this rule.):
     * feedback (suggestion):
     * Create a quantity
     * explanation
     * The quantity Q is used in the causal model builder. Maybe
     * this quantity should be defined as a quantity type in the
     * quantity builder (notice, that only quantity types should

```

```

    * be defined in the quantity builder and not specific
    * versions of that quantity type).
    */

    // NOT IMPLEMENTED

    /**
     * global: wrt model Fragment Builder and Scenario Builder
     * id: QBG2
     * version: 1.0
     * rule content (heuristic):
     * For each quantity Q that exists in the quantity builder and
     * that is not used in any model fragment and not used in any
     * scenario:
     * feedback (suggestion):
     * Unused quantity
     * explanation
     * The quantity Q is not used in any model fragment nor in any
     * scenario. Maybe this quantity is superfluous and can be
     * removed.
     */
    Rule(quantity_type q) {
        if {
            !in_mf(q) and !in_scenario(q)
        } then {
            advised(q);
        } title {
            Unused quantity ${q.name}
        } explanation {
            The quantity ${q.name} is not used in any model
            fragment nor in any scenario. Maybe this quantity is
            superfluous and can be removed.
        }
    }
}

```

A.3.5 The Quantity Space Builder

The Quantity Space Local Rule Base

```

RuleBase {
    use any, entity, model, attribute, model_fragment, qs, value_type;

    /**
     * local
     * id: QSBL1
     * version: 1.0
     * rule content (heuristic):
     * If no quantity spaces exist, expect for the default
     * quantity space mzp (NB: assuming that the builder default
     * provides the quantity space mzp, because GARP always needs
     * that quantity space). feedback (suggestion): Create a
     * quantity space
     * explanation
     * The quantity space builder is empty (expect for the default
     * space: mzp). No quantity spaces have been specified yet.
     * You can start by defining a quantity space.
     */
    Rule(model m) {
        if {
            !has_qss(m)
        } then {
            advised(m);
        } title {
            Create a quantity space
        } explanation {
            The quantity space builder is empty, expect for the
            default space: mzp. No quantity spaces have been
            specified yet. You can start by defining a quantity
            space.
        }
    }

    /**
     * local
     * id: QSBL2
     * version: 1.0
     * rule content (heuristic):
     * If one or more quantity spaces exist, in addition to the
     * default quantity space mzp (NB: assuming that in the
     * quantity builder always one quantity is selected, and that
     * the builder default provides the quantity space mzp,
     * because GARP always needs that quantity space).
     * feedback (suggestion):
     * Create or modify a quantity space
     * explanation
     * You have selected a quantity space. The following options
     * are possible: give the quantity space a different name,
     * change the values of the quantity space, delete the
     * quantity space, select an other quantity space, or create a
     * new quantity space.
     */
    Rule(model m) {
        if {
            has_qss(m)
        } then {
            advised(m);

```

```

    } title {
      Create or modify a quantity space
    } explanation {
      You have selected a quantity space. The following
      options are possible: give the quantity space a
      different name, change the values of the quantity
      space, delete the quantity space, select an other
      quantity space, or create a new quantity space.
    }
  }

/**
 * local
 * id: QSBL3
 * version: 1.0
 * rule content (fact):
 * For each quantity space (created by the model builder), the
 * set of value types should be a sequence of an alternating
 * points and interval
 * feedback (repair):
 * Invalid quantity space
 * explanation
 * Quantity space X is not a sequence of alternating points
 * and intervals. Change it, so that two adjacent values are
 * always of different types (either a point or an interval).
 */
Rule(qs qs) {
  if {
    !correct_value_order(qs)
  } then {
    warn(qs);
  } title {
    Invalid quantity space ${qs.name}.
  } explanation {
    Quantity space ${qs.name} is not a sequence of
    alternating points and intervals. Change it, so that
    two adjacent values are always of different types :
    either a point or an interval.
  }
}

/** QSBL4, 1.1 pre
 * This helper rule attempts to find a zero in the QS and if so,
 * assing a the has_zero predicate to the QS
 */
Rule(qs qs, value_type v) {
  if {
    has_value(v, qs)
    and (v.name=="Zero"
    or v.name=="zero")
  } then {
    has_zero(qs);
  }
}

/**
 * local
 * id: QSBL4
 * version: 1.1
 * rule content (heuristic):
 * For each quantity space (created by the model builder),
 * that does not include the point value zero.
 * feedback (suggestion):
 * Include point value zero
 * explanation (suggestion):
 * The quantity space X does not contain the point value zero.
 * Although correct, quantity spaces often contain the value
 * zero. Maybe you want to add the value zero to this quantity
 * space?
 */
Rule(qs qs) {
  if {
    !has_zero(qs)
  } then {
    advised(qs);
  } title {
    Include point value zero to ${qs.name}.
  } explanation {
    The quantity space ${qs.name} does not contain the
    point value zero. Although correct, quantity spaces
    often contain the value zero. Maybe you want to add
    the value zero to this quantity space?
  }
}

/**
 * local
 * id: QSBL5
 * version: 1.1
 * rule content (heuristic):
 * For each two quantity spaces, QS1 & QS2 (including default:
 * mzp), compare the value set. If the pairs have the same set
 * of value types (that is: use zero in the same way, starting
 * with the same lowest value type, have the same sequence of
 * intermediate types and have the same highest value, but
 * possible with different names).
 * feedback (suggestion):
 * Similar quantity spaces
 * explanation
 * The QS1 and QS2 have exactly the same set of qualitative
 * value types. This means that one the quantity spaces is in
 * principle superfluous. It can be deleted, although
 * insightful value names may be a reason to keep both
    * quantity spaces.
 */
Rule(qs a, qs b) {
  if {
    a.length == b.length
    and a.first == b.first
    and a.zero == b.zero
    and a != b
  } then {
    advised(a, b);
  } title {
    Similar quantity spaces ${a.name} and ${b.name}
  } explanation {
    The ${a.name} and ${b.name} have exactly the same
    set of qualitative value types. This means that one
    the quantity spaces is in principle superfluous. It
    can be deleted, although insightful value names may
    be a reason to keep both quantity spaces.
  }
}

/**
 * local
 * id: QSBL6
 * version: 1.1
 * rule content (heuristic):
 * For each quantity space Qsp if it contains the point value
 * zero, search for values above this value that include
 * 'min', 'minus', 'neg' or 'negative' (=X) as part of their
 * value names.
 * feedback (suggestion):
 * Incorrect value order
 * explanation
 * The label X usually refers to a value below zero. In quantity space Qsp
 * you use X as value name above zero. Maybe you want to change this, in
 * order to prevent confusion.
 */
Rule(qs q, value_type v) {
  if {
    has_value(v, q)
    and above_zero(v, q)
    and v.name ~= "pos|plus"
  } then {
    info(q,v);
  } title {
    Incorrect value order ${q.name}
  } explanation {
    The label ${v.name} usually refers to a value above
    zero. In quantity space ${q.name} you use ${v.name}
    as value name below zero. Maybe you want to change
    this, in order to prevent confusion.
  }
}

/**
 * local
 * id: QSBL7
 * version: 1.1
 * rule content (heuristic):
 * For each quantity space Qsp if it contains the point value
 * zero, search for values below this value that include
 * 'plus', 'pos', or 'positive' (=X) as part of their value
 * names.
 * feedback (suggestion):
 * Incorrect value order
 * explanation
 * The label X usually refers to a value above zero. In
 * quantity space Qsp you use X as value name below zero.
 * Maybe you want to change this, in order to prevent
 * confusion.
 */
Rule(qs q, value_type v) {
  if {
    has_value(v, q)
    and below_zero(v, q)
    and v.name ~= "min"
  } then {
    info(q,v);
  } title {
    Incorrect value order ${q.name}
  } explanation {
    The label ${v.name} usually refers to a value below
    zero. In quantity space ${q.name} you use ${v.name}
    as value name above zero. Maybe you want to change
    this, in order to prevent confusion.
  }
}

}

/**
 * global: wrt Quantity Builder
 * id: QSBG1
 * version: 1.0

```

The Quantity Space Cross Rule Base

```

RuleBase {
  use any, entity, model, attribute, model_fragment, object, quant;

  /**
   * global: wrt Quantity Builder
   * id: QSBG1
   * version: 1.0

```

```

* rule content (heuristic):
* For each quantity space Qsp that exists in the quantity
* space builder and that is not used by any quantity in the
* quantity builder: feedback (suggestion): Unused quantity
* space
* explanation
* The quantity space Qsp is not used by any quantity in the quantity
* builder. Maybe this quantity space is superfluous and can be removed. /**
*/
Rule(qs qs) {
  if {
    !used(qs)
  } then {
    advised(qs);
  } title {
    Unused quantity space ${qs.name}
  } explanation {
    The quantity space ${qs.name} is not used by any
    quantity in the quantity builder. Maybe this
    quantity space is superfluous and can be removed.
  }
}
}

```

A.3.6 The Model Fragment Builder

The Model Fragment Local Rule Base

```

RuleBase {
  use any, entity, structural_relation, attribute, model_fragment_type;
  define advised(any);

  /**
  * local
  * id: MFBL1
  * version: 1.0
  * rule content (fact):
  * If the model fragment does not have any conditional model
  * ingredients, and it does not have any consequential model
  * ingredients. explanation Model fragment is empty
  * feedback (fact):
  * The model fragment is empty. Nothing has been specified in
  * either the conditions nor the consequences. Maybe you want
  * to start filling the model fragment by adding an entity as
  * either a condition or a consequence. Or by adding an other
  * model fragment as a condition.
  */
  Rule(model_fragment_type mf) {
    if {
      !has_givens(mf)
      and !has_conditions(mf)
      and current(mf)
    } then {
      warn(mf);
    } title {
      Model Fragment ${mf.name} is empty.
    } explanation {
      The model fragment ${mf.name} is empty. Nothing has
      been specified in either the conditions nor the
      consequences. Maybe you want to start filling the
      model fragment by adding an entity as either a
      condition or a consequence. Or by adding an other
      model fragment as a condition.
    }
  }
}
/**
* local
* id: MFBL2
* version: 1.0
* rule content (heuristic):
* If the model fragment does not have any conditional model
* ingredients, but it does have some consequential model
* ingredients
* explanation
* There are no conditions
* feedback (suggestion):
* The model fragment has no conditions specified, that means
* that the knowledge specified in as a consequence will
* always apply. Is this what you want? Or do you want to
* specify specific conditions under which these facts should
* be applied.
*/
Rule(model_fragment_type mf) {
  if {
    has_givens(mf)
    and !has_conditions(mf)
    and current(mf)
  } then {
    advised(mf);
  } title {
    No conditions in ${mf.name}.
  } explanation {
    The model fragment ${mf.name} has no conditions
    specified, that means that the knowledge specified
    in as a consequence will always apply. Is this what
    you want? Or do you want to specify specific
    conditions under which these facts should be
    applied.
  }
}

* local
* id: MFBL3
* version: 1.0
* rule content (heuristic):
* If the model fragment does not have any consequential model
* ingredients, but it does have some conditional model
* ingredients
* explanation
* There are no consequences
* feedback (suggestion):
* The model fragment has no consequences specified, which
* means that no new knowledge will be added if the conditions
* are true. Is this what you want? Or do you want to specify
* some consequences that hold if the conditions are true.
*/
Rule(model_fragment_type mf) {
  if {
    !has_givens(mf)
    and has_conditions(mf)
    and current(mf)
  } then {
    advised(mf);
  } title {
    There are no consequences in ${mf.name}.
  } explanation {
    The model fragment ${mf.name} has no consequences
    specified, which means that no new knowledge will be
    added if the conditions are true. Is this what you
    want? Or do you want to specify some consequences
    that hold if the conditions are true.
  }
}

/**
* local
* id: MFBL4
* version: 1.0
* rule content (heuristic):
* For each existing entity E in the model fragment, either as
* condition or as consequence, that has not been given a
* quantity
* explanation
* Missing quantity
* feedback (suggestion):
* The entity E has not been given a quantity. A typical goal
* of creating a model fragment is to assign behavior to
* entities by means of quantities. Maybe, you want to give
* entity E a quantity.
*/
Rule(entity e, model_fragment_type mf) {
  if {
    in_mf(e, mf)
    and !has_quantity(e)
    and current(mf)
  } then {
    advised(e);
  } title {
    Missing quantity ${e.name}
  } explanation {
    The entity ${e.name} has not been given a quantity.
    A typical goal of creating a model fragment is to
    assign behavior to entities by means of quantities.
    Maybe, you want to give entity ${e.name} a quantity.
  }
}

/** MFBL5 1.0 - SUPPORT
* Support rule to support MFBL5. The rule finds out if two
* entities are:
* a) In the same model fragment, and the model fragment
* is the current selected
* b) and, connected by a structural relation
* c) and, the two entities are not the same one.
*/
Rule(entity e1, entity e2,
      structural_relation r, model_fragment_type mf) {
  if {
    in_mf(e1, mf)
    and in_mf(e2, mf)
    and connected(e1, r)
    and connected(e2, r)
    and current(mf)
    and e1 != e2
  } then {
    in_relation(e1, e2);
  }
}

/**
* local
* id: MFBL5
* version: 1.0
* rule content (fact):
* For either the conditions or the consequences, compare each
* entity E1 with each other entity E2: If there is no
* structural relation specified between E1 and E2.
* explanation

```

```

* Missing structural relation
* feedback (repair):
* The entities E1 and E2 are unrelated in this model
* fragment. This suggests that these entities are completely
* independent from each other. In general, this is not a
* desirable situation. Multiple entities are specified in
* model fragments to define the behavior resulting from their
* interaction. Probably, you have to add a structural
* relation that specifies the relationship between these two
* entities.
*/
Rule(entity e1, entity e2, model_fragment_type mf) {
  if {
    !in_relation(e1, e2) and e1 != e2
    and in_mf(e1, mf)
    and in_mf(e2, mf)
    and current(mf)
  } then {
    advised(e1, e2);
  } title {
    Missing structural relation ${e1.name}
  } explanation {
    The entities ${e1.name} and ${e2.name} are unrelated
    in this model fragment. This suggests that these
    entities are completely independent from each other.
    In general, this is not a desirable situation.
    Multiple entities are specified in model fragments
    to define the behavior resulting from their
    interaction. Probably, you have to add a structural
    relation that specifies the relationship between
    these two entities.
  }
}

/**
* local
* id: MFBL6
* version: 1.1
* rule content (heuristic):
* OLD version 1.0: For either the conditions or the
* consequences, Compare the quantities of a single entity
* with each other (by pairs, thus E1/Q1 with E1/Q2, etc.): If
* there is no behavioral dependency specified between E1/Q1
* and E1/Q2. (NB: assuming that quantities always have a
* quantity space). NEW version 1.1: as version 1.0 but only
* consider quantities of an entity that have NO dependency
* with any of the other quantities of that entity (in other
* words: each Q of an entity should be related to at least
* one other quantity of that entity).
* explanation
* Missing behavioral dependency
* feedback (suggestion):
* The quantities Q1 and Q2 of entity E1 are unrelated (expect
* for both belonging to the same entity). Often quantities of
* a single entity have behavioral relationships, specifying
* their dependency. Maybe you want to add a dependency
* between these quantities.
*/
Rule(entity e, quantity q1, quantity q2, model_fragment_type mf) {
  if {
    connected(q1, e)
    and connected(q2, e)
    and q1 != q2
    and !related(q1, q2)
    and current(mf)
    and in_mf(q1, mf)
    and in_mf(q2, mf)
  } then {
    advised(q1, q2, e);
  } title {
    Missing behavioral dependency between ${q1.name}
    and ${q2.name}
  } explanation {
    The quantities ${q1.name} and ${q2.name} of entity
    ${e.name} are unrelated, expect for both belonging
    to the same entity. Often quantities of a single
    entity have behavioral relationships, specifying
    their dependency. Maybe you want to add a dependency
    between these quantities.
  }
}

/** MFBL7 1.1 - SUPPORT
* This rule is a support rule for MFBL7. The rule tries to
* find out if two entities are related. Related in this
* context means that they have quantities that are connected
* by a dependency.
*/
Rule(entity e1, entity e2, quantity q1, quantity q2, model_fragment_type mf) {
  if {
    connected(q1, e1)
    and connected(q2, e2)
    and e1 != e2
    and related(q1, q2)
    and current(mf)
  } then {
    related(e1, e2);
  }
}

/**
* local
* id: MFBL7
* version: 1.0
* rule content (heuristic):
* For both the conditions or the consequences (no distinction
* needed), compare all the quantities of two entities to each
* other (by pairs, thus E1/Q1y-Q1x with E2/Q2y-Q2x, etc.): If
* there is no behavioral dependency specified between any of
* the quantities of the two entities (NB: assuming that
* quantities always have a quantity space).
* explanation
* Missing behavioral dependency
* feedback (suggestion):
* The quantities of E1 and E2 are unrelated. Often quantities
* of different entities have behavioral relationships,
* specifying their dependency. Maybe you want to add a
* dependency between some of the quantities of these
* different entities.
*/
Rule(entity e1, entity e2, model_fragment_type mf) {
  if {
    !related(e1, e2)
    and in_mf(e1, mf)
    and in_mf(e2, mf)
    and e1 != e2
    and current(mf)
  } then {
    advised(e1, e2);
  } title {
    Missing behavioural dependency between ${e1.name}
    and ${e2.name}
  } explanation {
    The quantities of ${e1.name} and ${e2.name} are
    unrelated. Often quantities of different entities
    have behavioural relationships, specifying their
    dependency. Maybe you want to add a dependency
    between some of the quantities of these different
    entities.
  }
}

/**
* local
* id: MFBL8
* version: 1.0
* rule content (heuristic):
* If one entity is selected (and nothing else) (either as
* condition or consequence, no distinction needed)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected an entity. The following options are
* possible: add a quantity to the entity, add an attribute to
* the entity, give the entity a different local name, change
* the supertype of the entity (is this possible?), delete the
* selected entity, or undo your selection.
*/
Rule(drawing d, entity e) {
  if {
    d.selection=="1"
    and selected(e)
  } then {
    advised(e);
  } title {
    Work on the current selection
  } explanation {
    You have selected an entity ${e.name}. The following
    options are possible: add a quantity to the entity,
    add an attribute to the entity, give the entity a
    different local name, delete the selected entity, or
    undo your selection.
  }
}

/**
* local
* id: MFBL9
* version: 1.0
* rule content (heuristic):
* If one attribute is selected (and nothing else) (either as
* condition or consequence, no distinction needed)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected an attribute. The following options are
* possible: change the value of the attribute, delete the
* selected attribute, or undo your selection.
*/
Rule(drawing d, attribute a) {
  if {
    d.selection=="1"
    and selected(a)
  } then {
    advised(a);
  } title {
    Work on the current selection
  } explanation {
    You have selected an attribute ${a.name}. The
    following options are possible: change the value of
    the attribute, delete the selected attribute, or
    undo your selection.
  }
}

/**
* local
* id: MFBL10
* version: 1.0

```

```

* rule content (heuristic):
* If one structural relation is selected (and nothing else)
* (either as condition or consequence, no distinction needed)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected a structural relation. The following
* options are possible: change the LHS and/or RHS entity of
* the relation, delete the selected relation, or undo your
* selection.
*/
Rule(drawing d, structural_relation s) {
  if {
    d.selection=="1"
    and selected(s)
  } then {
    advised(s);
  } title {
    Work on the current selection
  } explanation {
    You have selected a structural relation ${s.name}.
    The following options are possible: change the LHS
    and-or RHS entity of the relation, delete the
    selected relation, or undo your selection.
  }
}

/**
* local
* id: MFBL11
* version: 1.0
* rule content (heuristic):
* If one quantity is selected (and nothing else) (either as
* condition or consequence, no distinction needed)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected a quantity. The following options are
* possible: change the value and/or the derivative assigned
* to the quantity, give the quantity a different local name,
* delete the selected quantity, or undo your selection.
*/
Rule(drawing d, quantity q) {
  if {
    d.selection=="1"
    and selected(q)
  } then {
    advised(q);
  } title {
    Work on the current selection
  } explanation {
    You have selected a quantity ${q.name}. The
    following options are possible: change the value
    and-or the derivative assigned to the quantity, give
    the quantity a different local name, delete the
    selected quantity, or undo your selection.
  }
}

/**
* local
* id: MFBL12
* version: 1.0
* rule content (heuristic):
* If one supertype model fragment is selected (and nothing
* else) (default a condition)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected a supertype model fragment. The following
* options are possible: change the way in which the entities
* from the supertype model fragment are re-used in the
* current model fragment, delete the selected supertype model
* fragment, or undo your selection.
*/
Rule(drawing d, model_fragment_type mft, model_fragment_type mf) {
  if {
    d.selection=="1"
    and selected(mft)
    and supertype_of(mf, mft)
  } then {
    advised(mft);
  } title {
    Work on the current selection
  } explanation {
    You have selected a supertype model fragment
    ${mft.name}. The following options are possible:
    change the way in which the entities from the
    supertype model fragment are re-used in the current
    model fragment, delete the selected supertype model
    fragment, or undo your selection.
  }
}

/**
* local
* id: MFBL13
* version: 1.0
* rule content (heuristic):
* If one conditional model fragment is selected (and nothing
* else) (default a condition)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected a condition model fragment. The following
* options are possible: change the way in which the entities
* from the condition model fragment are re-used in the
* current model fragment, delete the selected conditional
* model fragment, or undo your selection.
*/
Rule(drawing d, model_fragment_type mft, model_fragment_type mf) {
  if {
    d.selection=="1"
    and selected(mft)
    and !supertype_of(mf, mft)
  } then {
    advised(mft);
  } title {
    Work on the current selection
  } explanation {
    You have selected a condition model fragment
    ${mft.name}. The following options are possible:
    change the way in which the entities from the
    condition model fragment are re-used in the current
    model fragment, delete the selected conditional
    model fragment, or undo your selection.
  }
}

/**
* local
* id: MFBL14
* version: 1.0
* rule content (suggestion):
* If two entities are selected (and nothing else) (BOTH as
* either condition or consequence)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected two entities. You can add a structural
* relation between the selected entities or undo your
* selection.
*/
Rule(drawing d, entity e1, entity e2) {
  if {
    d.selection=="2"
    and selected(e1)
    and selected(e2)
    and e1 != e2
  } then {
    advised(e1, e2);
  } title {
    Work on the current selection
  } explanation {
    You have selected two entities, ${e1.name} and
    ${e2.name}. You can add a structural relation
    between the selected entities or undo your
    selection.
  }
}

/**
* local
* id: MFBL15
* version: 1.0
* rule content (suggestion):
* If two quantities are selected (and nothing else) (either
* as condition or consequence, no distinction needed)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected two quantities. You can add a behavioral
* dependency (a (in)equality, proportionality or influence)
* between the selected quantities or undo your selection.
*/
Rule(drawing d, quantity q1, quantity q2) {
  if {
    d.selection=="2"
    and selected(q1)
    and selected(q2)
    and q1 != q2
  } then {
    advised(q1, q2);
  } title {
    Work on the current selection
  } explanation {
    You have selected two quantities, ${q1.name} and
    ${q2.name}. You can add a behavioural dependency (a
    (in)equality, proportionality or influence) between
    the selected quantities or undo your selection.
  }
}

/**
* local
* id: MFBL16
* version: 1.0
* rule content (suggestion):
* If two quantity spaces are selected (and nothing else)
* (either as condition or consequence, no distinction needed)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected two quantity spaces. You can add a
* quantity correspondence between the selected quantity
* spaces or undo your selection.
*/
Rule(drawing d, qs qs1, qs qs2) {
  if {

```



```

        d.selection=="2"
        and selected(qs1)
        and selected(qs2)
        and qs1 != qs2

    } then {
        advised(qs1, qs2);
    } title {
        Work on the current selection
    } explanation {
        You have selected two quantity spaces, ${qs1.name}
        and ${qs2.name}. You can add a quantity
        correspondence between the selected quantity spaces
        or undo your selection.
    }
}

/**
 * ID: MFBL17 1.0 - SUPPORT 1
 * This support rule identifies a value of type point.
 */
Rule(value v, value_type type) {
    if {
        of_type(v, type)
        and point(type)
    } then {
        point(v);
    }
}

/**
 * ID: MFBL17 1.0 - SUPPORT 2
 * This support rule identifies a value of type interval.
 */
Rule(value v, value_type type) {
    if {
        of_type(v, type)
        and interval(type)
    } then {
        interval(v);
    }
}

/**
 * ID: MFBL17 1.0 - SUPPORT 3
 * This support rule identifies any set of two values that are
 * points and assigns the predicate 'point' over them.
 */
Rule(value v1, value v2) {
    if {
        v1 != v2
        and point(v1)
        and point(v2)
    } then {
        points(v1, v2);
    }
}

/**
 * local
 * id: MFBL17
 * version: 1.0
 * rule content (suggestion):
 * If two values are selected (and nothing else) (either point
 * or interval values, no distinction needed) (either as
 * condition or consequence, no distinction needed)
 * feedback (suggestion):
 * Work on the current selection
 * explanation
 * You have selected two values. You can add a value
 * correspondence between the selected values or undo your
 * selection.
 */
Rule(drawing d, value v1, value v2) {
    if {
        d.selection=="2"
        and selected(v1)
        and selected(v2)
        and v1 != v2
        and !points(v1, v2)
    } then {
        advised(v1, v2);
    } title {
        Work on the current selection
    } explanation {
        You have selected two values, ${v1.name} and
        ${v2.name}. You can add a value correspondence
        between the selected values or undo your selection.
    }
}

/**
 * local
 * id: MFBL18
 * version: 1.0
 * rule content (suggestion):
 * If two point values are selected (and nothing else) (BOTH
 * points values) (either as condition or consequence, no
 * distinction needed)
 * feedback (suggestion):
 * Work on the current selection
 * explanation
 * You have selected two point values. You can add an
 * (in)equality between the selected point values or undo your
 * selection.
 */
Rule(drawing d, value v1, value v2) {
    if {
        d.selection=="2"
        and selected(v1)
        and selected(v2)
        and v1 != v2
        and points(v1, v2)
    } then {
        advised(v1, v2);
    } title {
        Work on the current selection
    } explanation {
        You have selected two point values, ${v1.name} and
        ${v2.name}. You can add an (in)equality between the
        selected point values or undo your selection.
    }
}

/**
 * ID: MFBL19 1.0 - SUPPORT 1.
 * This support method searches for entities in the structural hierarchy that
 * are peers, that is, have the same parent:
 *
 * parent
 * / \
 * p   p
 */
Rule(entity_type a, entity_type b, entity_type parent) {
    if {
        parent.name != "nil"
        and subtype_of(a, parent)
        and subtype_of(b, parent)
        and a != b
    } then {
        peer(a,b)
    } title {
    } explanation {
    }
}

/**
 * ID: MFBL19 1.0 - SUPPORT 2.
 * This support rule adds the peer predicate of the prior
 * support rule (MFBL19 1.0 - SUPPORT 1) to the entities in
 * the current model fragment.
 */
Rule(entity_type a, entity_type b,
      entity e1, entity e2, model_fragment_type mf ) {
    if {
        peer(a,b)
        and of_type(e1, a)
        and of_type(e2, b)

        and current(mf)
        and in_mf(e1, mf)
        and in_mf(e2, mf)

        and e1 != e2
        and a != b
    } then {
        peer(e1,e2)
    } title {
    } explanation {
    }
}

/**
 * ID: MFBL19 1.0 - SUPPORT 3.
 * This support rule simply adds the same_type predicate to
 * all the quantities with the same type.
 */
Rule(quantity_type type, quantity e1, quantity e2 ) {
    if {
        of_type(e1, type)
        and of_type(e2, type)

        and e1 != e2
    } then {
        same_type(e1,e2)
    } title {
    } explanation {
    }
}

/**
 * ID: MFBL19 1.0 - SUPPORT 4.
 * This support rule adds the equal_quantity predicate to all
 * pairs of entities that both have a same type of quantity.
 */
Rule(entity e1, entity e2, quantity q1, quantity q2) {
    if {
        peer(e1, e2)
        and same_type(q1, q2)
        and connected(q1, e1)
        and connected(q2, e2)

        and e1 != e2
        and q1 != q2
    }
}

```

```

    } then {
        equal_quantity(e1,q1)
        equal_quantity(e2,q2)
    } title {
    } explanation {
    }
}

/**
 * local
 * id: MFBLL17
 * version: 1.1
 * rule content (heuristic):
 * If each two entities E1 and E2 in the current model
 * fragment that are both immediate subtypes of the same
 * supertype E3, find each quantity occurrence (=User Given
 * Name) of quantity type Q that is assigned to E1 but not to
 * E2 (thus this rule should NOT fire when both entities have
 * the same set of quantities (occurrences) and quantity
 * types).
 * explanation
 * Make entities similar
 * feedback (suggestion):
 * The quantity 'User Given Name' of quantity type Q is
 * assigned to E1, but a similar quantity is not assigned to
 * E2, even though E1 and E2 are similar type of entities with
 * probably similar behaviour.
 */
Rule(entity e, entity other, quantity q, model_fragment_type mf ) {
    if {
        !equal_quantity(e,q)
        and connected(q, e)
        and peer(e, other)

        and current(mf)
        and in_mf(e, mf)
        and in_mf(q, mf)
    } then {
        advised(e,other, q)
    } title {
        Make entities ${e.name} and ${other.name} similar.
    } explanation {
        The quantity of quantity type ${q.name} is assigned
        to ${e.name}, but a similar quantity is not assigned
        to ${other.name}, even though ${e.name} and
        ${other.name} are similar type of entities with
        probably similar behavior.
    }
}
}

```

The Model Fragment Cross Rule Base

```

RuleBase {
    use any, entity_type, structural_relation_type,
        attribute_type, model_fragment_type;

    /**
     * ID: MFBG1 1.0 - SUPPORT 1
     * This rule puts all children of a parent entity_type
     * in the model fragment.
     */
    Rule(entity_type child, entity_type parent) {
        if {
            parent.name != "nil"
            and subtype_of(child, parent)
            and in_mf(parent)
        } then {
            in_mf(child)
        }
    }

    /**
     * ID: MFBG1 1.0 - SUPPORT 2
     * This rule puts the parent of a child entity_type in the
     * model fragment.
     */
    Rule(entity_type child, entity_type parent) {
        if {
            parent.name != "nil"
            and subtype_of(child, parent)
            and in_mf(child)
        } then {
            in_mf(parent)
        }
    }
}

/**
 * global: wrt Structural Builder
 * id: MFBG1
 * version: 1.0
 * rule content (heuristic):
 * For each entity E that exists in the structural builder and
 * that is not used in any model fragment: (A more refined
 * version of this rule would not take any entity but only
 * entities that do not have a supertype entity that is used
 * in any model fragment. A second optimization would not
 * mention all those entities, but only the highest supertype
 * below 'nil')
 */
    feedback (suggestion):
    * Unused entity
    * explanation
    * The entity E is not used in any model fragment. Maybe you
    * want to include it in this model fragment.
    */
    Rule(entity_type e, model_fragment_type mf) {
        if {
            !in_mf(e) and !e.name=="nil"
        } then {
            advised(e);
        } title {
            add ${e.name} to a model fragment
        } explanation {
            Entity ${e.name} is not yet used in any model
            fragment, it does not make sense to have an entity
            in your model and not make use of it in a Model
            Fragment.
        }
    }

    /**
     * global: wrt Structural Builder
     * id: MFBG3
     * version: 1.0
     * rule content (heuristic):
     * If entity E is used in the active model fragment, and this
     * entity has one or more attributes A assigned to it in the
     * structural builder, then for each attribute A that is not
     * used in any model fragment:
     * feedback (suggestion):
     * Unused attribute
     * explanation
     * The attribute A is not used in any model fragment. Maybe
     * you want to include it in this model fragment.
     */
    Rule(attribute_type a, model_fragment_type mf, entity_type e) {
        if {
            !in_mf(a) // relation not in mf
            // mf is the current mf
            and current(mf)
            // relation has a connection which
            // is in mf.. see last condition...
            and connected(e, a)
            // connection in mf
            and in_mf(e, mf)
        } then {
            advised(a);
            a.target = e.name;
        } title {
            Add ${a.name} to entity ${a.target}
        } explanation {
            Attribute ${a.name} from ${a.target} is not used in
            the model, it does not make sense to create an
            attribute and not use it in a model fragment.
        }
    }

    /**
     * global: wrt Structural Builder
     * id: MFBG4
     * version: 1.0
     * rule content (heuristic):
     * If two entities E1 and E2 are used in the active model
     * fragment, and between E1 and E2 one or more structural
     * relations SR have been defined in the structural builder,
     * then for each structural relation SR that is not used in
     * any model fragment:
     * feedback (suggestion):
     * Unused structural relation
     * explanation
     * The structural relation SR is not used in any model
     * fragment. Maybe you want to include it in this model
     * fragment.
     */
    Rule(structural_relation_type s, model_fragment_type mf,
        entity_type e1, entity_type e2) {
        if {
            // relation not in mf
            !in_mf(s)
            // mf is the current mf
            and current(mf)
            // relation has a lhs which is in mf..
            // see last 2 conditions...
            and lhs(e1, s)
            // relation has a rhs which is in mf..
            // see last 2 conditions...
            and rhs(e2, s)
            // lhs in mf
            and in_mf(e1, mf)
            // rhs in mf..
            and in_mf(e2, mf)
        } then {
            advised(s);
            s.target = e1.name
            s.target2 = e2.name
        } title {
            Add ${s.name} to entity_type ${s.target} and
            entity_type ${s.target2}
        } explanation {
            Structural relation ${s.name} is not used in any
            model fragment. It does not make sense to make a
            relation and not use it in any model fragment.
        }
    }
}

```

```

/**
 * global: wrt Quantity Builder
 * id: MFBG5
 * version: 1.0
 * rule content (heuristic):
 * For each quantity Q that exists in the quantity builder and that is not
 * used in any model fragment:
 * feedback (suggestion):
 * Unused quantity
 * explanation
 * The quantity Q is not used in any model fragment. Maybe you
 * want to include it in this model fragment.
 */
Rule(quantity_type q, model_fragment_type mf) {
  if {
    !in_mf(q)
    and current(mf)
  } then {
    advised(q, mf);
  }
  title {
    Add quantity ${q.name} to model fragment ${mf.name}
  }
  explanation {
    The quantity ${q.name} is not used in any model
    fragment. Maybe you want to include it in this model
    fragment.
  }
}

/**
 * global: wrt Structural Builder
 * id: MFBG6
 * version: 1.0
 * rule content (heuristic):
 * If entity E is used in the active model fragment, and this
 * entity has one or more attributes A assigned to it in the
 * structural builder, then for each VALUE V (of attribute A)
 * that is not used in any model fragment (notice, this rules
 * relates to MFBG3, but fires in slightly more specific
 * situations, if working correct MFBG3 and MFBG6 should
 * should not fire together).
 * feedback (suggestion):
 * Unused attribute value
 * explanation
 * The value V of attribute A is not used in any model
 * fragment. Maybe you want to include it in this model
 * fragment.
 */
Rule(attribute_value av, attribute_type at, model_fragment_type mf, entity_type e) {
  if {
    !in_mf(av)
    and in_mf(at)
    and owner(av, at)
    // mf is the current mf
    and current(mf)
    // relation has a connection which is in mf..
    // see last condition...
    and connected(e, at)
    // connection in mf
    and in_mf(e, mf)
  } then {
    advised(av, at);
  }
  title {
    Unused attribute value ${av.name} from ${at.name}
  }
  explanation {
    The value ${av.name} of attribute ${at.name} is not
    used in any model fragment. Maybe you want to
    include it in this model fragment.
  }
}

}

}

} title {
  Scenario is empty ${s.name}.
}
explanation {
  The scenario ${s.name} is empty. Nothing has been
  specified yet. You can start filling the scenario by
  adding an entity.
}

/**
 * local
 * id: ScBL2
 * version: 1.0
 * rule content (heuristic):
 * For each existing entity E in the scenario that has not
 * been given a quantity
 * explanation
 * Missing quantity
 * feedback (suggestion):
 * The entity E has not been given a quantity. This may result
 * in the simulation not being able to derive certain
 * behaviour. Maybe, you want to give entity E a quantity.
 */
Rule(entity e, scenario_type sc) {
  if {
    in_scenario(e, sc)
    and !has_quantity(e)
    and current(sc)
  } then {
    advised(e);
  }
  title {
    Missing quantity ${e.name}
  }
  explanation {
    The entity ${e.name} has not been given a quantity.
    This may result in the simulation not being able to
    derive certain behavior. Maybe, you want to give
    entity ${e.name} a quantity.
  }
}

/**
 * local
 * id: ScBL3
 * version: 1.0
 * rule content (heuristic):
 * If one or more quantities exist (NB: assuming that
 * quantities always have a quantity space) and none of them
 * has been assigned a value
 * explanation
 * No initial values
 * feedback (suggestion):
 * None of the quantities has a been given an initial value.
 * This may result in the simulation not being able to derive
 * any behaviour. Maybe, you want to give a quantity a specific
 * value.
 */
Rule(quantity q, scenario_type sc) {
  if {
    in_scenario(q, sc)
    and !has_value(q);
  } then {
    info(q);
    //in_relation(e2);
  }
  title {
    No initial values ${q.name}
  }
  explanation {
    The quantity ${q.name} has not been given an initial
    value. This may result in the simulation not being
    able to derive any behavior. Maybe, you want to give
    a quantity a specific value.
  }
}

}

}

}

/**
 * ID: ScBL4 1.0 SUPPORT
 * This rule checks if two entities are connected by a
 * structural relation.
 */
Rule(entity e1, entity e2, structural_relation r, scenario_type sc) {
  if {
    in_scenario(e1, sc)
    and in_scenario(e2, sc)
    and in_scenario(r, sc)
    and connected(e1, r)
    and connected(e2, r)
    and current(sc)
    and e1 != e2
    and in_scenario(r, sc)
  } then {
    in_relation(e1, e2);
  }
}

/**
 * local
 * id: ScBL4
 * version: 1.0
 * rule content (fact):
 * Compare each entity E1 with each other entity E2: If there
 * is no structural relation specified between E1 and E2
 * explanation
 * Missing structural relation
 * feedback (repair):

```

A.3.7 The Scenario Builder

The Scenario Local Rule Base

```

RuleBase {

  use any, entity, structural_relation, attribute, scenario;
  define advised(any);

  /**
   * local
   * id: ScBL1
   * version: 1.0
   * rule content (fact):
   * If the scenario does not have any model ingredients.
   * explanation
   * Scenario is empty
   * feedback (fact):
   * The scenario is empty. Nothing has been specified yet. You
   * can start filling the scenario by adding an entity.
   */
  Rule(scenario_type s) {
    if {
      !has_content(s)
      and current(s)
    } then {
      warn(s);
    }
  }
}

```

```

* The entities E1 and E2 are unrelated in this scenario. This
* suggests that these entities are completely independent
* from each other. In general, this is not a desirable
* situation. Probably, you have to add a structural relation
* that specifies the relationship between these two entities.
*/
Rule(entity e1, entity e2, scenario_type sc) {
  if {
    !in_relation(e1, e2) and e1 != e2
    and in_scenario(e1, sc)
    and in_scenario(e2, sc)
    and current(sc)
  } then {
    advised(e1, e2);
  } title {
    Missing structural relation ${e1.name}
  } explanation {
    The entities ${e1.name} and ${e2.name} are unrelated
    in this scenario. This suggests that these entities
    are completely independent from each other. In
    general, this is not a desirable situation.
    Probably, you have to add a structural relation that
    specifies the relationship between these two
    entities.
  }
}

/**
* local
* id: ScBL5
* version: 1.0
* rule content (heuristic):
* Compare each entity/quantity E1/Q1 with each other
* entity/quantity E2/Q2: If E1 and E2 are of the same type,
* and if Q1 and Q2 are of the same type, and there is no
* (in)equality dependency specified between Q1 and Q2 (NB:
* assuming that quantities always have a quantity space).
* explanation
* Missing inequality
* feedback (suggestion):
* The entities and quantities E1/Q1 and E2/Q2 are of similar
* type, but they are unrelated in this scenario. In
* scenario's, inequality dependencies are often specified
* between similarly looking quantities in order to
* investigate specific behaviours. So, maybe you want to add
* an inequality statement between these quantities.
*/
Rule(entity e1, entity e2, quantity q1, quantity q2,
      scenario_type sc, entity_type et, quantity_type qt) {
  if {
    connected(q1, e1)
    and connected(q2, e2)
    and e1 != e2
    and !related(q1, q2)
    and current(sc)
    and of_type(e1, et)
    and of_type(e2, et)
    and of_type(q1, qt)
    and of_type(q2, qt)
    and in_scenario(q1, sc)
    and in_scenario(q2, sc)
    and in_scenario(e1, sc)
    and in_scenario(e2, sc)
  } then {
    //related(e1, e2, q1, q2);
    //warn(e1, q2);
    advised(e1, e2, q1, q2);
  } title {
    Missing inequality ${e1.name} and ${q1.name}
  } explanation {
    The entities and quantities ${e1.name} - ${q1.name}
    and ${e2.name} - ${q2.name} are of similar type, but
    they are unrelated in this scenario. In scenario s,
    inequality dependencies are often specified between
    similarly looking quantities in order to investigate
    specific behaviours. So, maybe you want to add an
    inequality statement between these quantities.
  }
}

/**
* local
* id: ScBL6
* version: 1.0
* rule content (heuristic):
* If one entity is selected (and nothing else)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected an entity. The following options are
* possible: add a quantity to the entity, add an attribute to
* the entity, give the entity a different local name, change
* the supertype of the entity (is this possible?), delete the
* selected entity, or undo your selection.
*/
Rule(drawing d, entity e) {
  if {
    d.selection=="1"
    and selected(e)
  } then {
    advised(e);
  } title {
    Work on the current selection
  } explanation {
    You have selected an entity ${e.name}. The following
    options are possible: add a quantity to the entity,
    add an attribute to the entity, give the entity a
    different local name, change the supertype of the
    entity (is this possible?), delete the selected
    entity, or undo your selection.
  }
}

/**
* local
* id: ScBL7
* version: 1.0
* rule content (heuristic):
* If one attribute is selected (and nothing else)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected an attribute. The following options are
* possible: change the value of the attribute, delete the
* selected attribute, or undo your selection.
*/
Rule(drawing d, attribute a) {
  if {
    d.selection=="1"
    and selected(a)
  } then {
    advised(a);
  } title {
    Work on the current selection
  } explanation {
    You have selected an attribute ${a.name}. The
    following options are possible: change the value of
    the attribute, delete the selected attribute, or
    undo your selection.
  }
}

/**
* local
* id: ScBL8
* version: 1.0
* rule content (heuristic):
* If one structural relation is selected (and nothing else)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected a structural relation. The following
* options are possible: change the LHS and/or RHS entity of
* the relation, delete the selected relation, or undo your
* selection.
*/
Rule(drawing d, structural_relation s) {
  if {
    d.selection=="1"
    and selected(s)
  } then {
    advised(s);
  } title {
    Work on the current selection
  } explanation {
    You have selected a structural relation ${s.name}.
    The following options are possible: change the LHS
    and-or RHS entity of the relation, delete the
    selected relation, or undo your selection.
  }
}

/**
* local
* id: ScBL9
* version: 1.0
* rule content (heuristic):
* If one quantity is selected (and nothing else)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected a quantity. The following options are
* possible: change the value and/or the derivative assigned
* to the quantity, give the quantity a different local name,
* delete the selected quantity, or undo your selection.
*/
Rule(drawing d, quantity q) {
  if {
    d.selection=="1"
    and selected(q)
  } then {
    advised(q);
  } title {
    Work on the current selection
  } explanation {
    You have selected a quantity ${q.name}. The
    following options are possible: change the value
    and-or the derivative assigned to the quantity, give
    the quantity a different local name, delete the
    selected quantity, or undo your selection.
  }
}

/**
* local
* id: ScBL10
* version: 1.0
* rule content (suggestion):
* If two entities are selected (and nothing else)

```

```

* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected two entities. You can add a structural
* relation between the selected entities or undo your
* selection.
*/
Rule(drawing d, entity e1, entity e2) {
  if {
    d.selection=="2"
    and selected(e1)
    and selected(e2)
    and e1 != e2
  } then {
    advised(e1, e2);
  } title {
    Work on the current selection
  } explanation {
    You have selected two entities, ${e1.name} and
    ${e2.name}. You can add a structural relation
    between the selected entities or undo your
    selection.
  }
}

/**
* local
* id: ScBL11
* version: 1.0
* rule content (suggestion):
* If two quantities are selected (and nothing else)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected two quantities. You can add an
* (in)equality between the selected quantities or undo your
* selection.
*/
Rule(drawing d, quantity q1, quantity q2) {
  if {
    d.selection=="2"
    and selected(q1)
    and selected(q2)
    and q1 != q2
  } then {
    advised(q1, q2);
  } title {
    Work on the current selection
  } explanation {
    You have selected two quantities, ${q1.name} and
    ${q2.name}. You can add (in)equality between the
    selected quantities or undo your selection.
  }
}

/**
* ID: ScBL12 1.0 - SUPPORT 2
* This support rule identifies a value of type point.
*/
Rule(value v, value_type type) {
  if {
    of_type(v, type)
    and point(type)
  } then {
    point(v);
  }
}

/**
* ID: ScBL12 1.0 - SUPPORT 2
* This support rule identifies any set of two values that are
* points and assigns the predicate 'point' over them.
*/
Rule(value v1, value v2) {
  if {
    v1 != v2
    and point(v1)
    and point(v2)
  } then {
    points(v1, v2);
  }
}

/**
* local
* id: ScBL12
* version: 1.0
* rule content (suggestion):
* If two point values are selected (and nothing else) (BOTH
* points values)
* feedback (suggestion):
* Work on the current selection
* explanation
* You have selected two point values. You can add an
* (in)equality between the selected point values or undo your
* selection.
*/
Rule(drawing d, value v1, value v2) {
  if {
    d.selection=="2"
    and selected(v1)
    and selected(v2)
    and v1 != v2
    and points(v1, v2)

```

```

  } then {
    advised(v1, v2);
  } title {
    Work on the current selection
  } explanation {
    You have selected two point values, ${v1.name} and
    ${v2.name}. You can add an (in)equality between the
    selected point values or undo your selection.
  }
}
}
}

```

The Scenario Cross Rule Base

```

RuleBase {
  use any, entity_type, structural_relation_type,
        attribute_type, scenario_type;
  define advised(any);

  /** ID: MFBG1 1.0 - SUPPORT */
  Rule(entity_type child, entity_type parent) {
    if {
      parent.name != "nil"
      and subtype_of(child, parent)
      and in_mf(parent)
    } then {
      in_mf(child)
    } title {
    } explanation {
    }
  }

  /** ID: MFBG1 1.0 - SUPPORT */
  Rule(entity_type child, entity_type parent) {
    if {
      parent.name != "nil"
      and subtype_of(child, parent)
      and in_scenario(child)
    } then {
      in_scenario(parent)
    } title {
    } explanation {
    }
  }

  /** ID: MFBG1 1.0 - SUPPORT */
  Rule(entity_type e, scenario_type mf) {
    if {
      //in_isa(e) and (!in_scenario(e))
      !in_scenario(e) and !e.name=="nil"
    } then {
      advised(e);
    } title {
      add ${e.name} to a model fragment
    } explanation {
      Entity ${e.name} is not yet used in any model
      fragment, it does not make sense to have an entity
      in your model and not make use of it in a Model
      Fragment.
    }
  }

  /** MFBG3 1.0 */
  Rule(attribute_type a, scenario_type mf, entity_type e) {
    if {
      // relation not in mf
      !in_scenario(a)
      // mf is the current mf
      and current(mf)
      // relation has a connection which is in mf..
      // see last condition...
      and connected(e, a)
      // connection in mf
      and in_scenario(e, mf)
    } then {
      advised(a);
      a.target = e.name;
    } title {
      Add ${a.name} to entity ${a.target}
    } explanation {
      Attribute ${a.name} from ${a.target} is not used in
      the model, it does not make sense to create an
      attribute and not use it in a model fragment.
    }
  }

  /** MFBG4 1.0 */
  Rule(structural_relation_type s, scenario_type mf, entity_type e1, entity_type e2) {
    if {
      // relation not in mf
      !in_scenario(s)
      // mf is the current mf
      and current(mf)
      // relation has a lhs which is in mf..
      // see last 2 conditions...
      and lhs(e1, s)

```

```

        // relation has a rhs which is in mf..
        // see last 2 conditions...
        and rhs(e2, s)
        // lhs in mf
        and in_scenario(e1, mf)
        // rhs in mf..
        and in_scenario(e2, mf)
    } then {
        advised(s);
        s.target = e1.name
        s.target2 = e2.name
    } title {
        Add ${s.name} to entity_type ${s.target} and
        entity_type ${s.target2}
    }
    explanation {
        Structural relation ${s.name} is not used in any
        model fragment. It does not make sense to make a
        relation and not use it in any model fragment.
    }
}

/** MFBG5 1.0 */
Rule(quantity_type q, scenario_type mf) {
    if {
        !in_scenario(q)
        and current(mf)
    } then {
        advised(q, mf);
    } title {
        Add quantity ${q.name} to model fragment ${mf.name}
    } explanation {
        The quantity ${q.name} is not used in any model
        fragment. Maybe you want to include it in this model
        fragment.
    }
}

/** MFBG6 1.0 */
Rule(attribute_value av, attribute_type at, scenario_type mf, entity_type e) {
    if {
        !in_scenario(av)
        and in_scenario(at)
        and owner(av, at)
        // mf is the current mf
        and current(mf)
        // relation has a connection which is in mf..
        // see last condition...
        and connected(e, at)
        // connection in mf
        and in_scenario(e, mf)
    } then {
        advised(av, at);
    } title {
        Unused attribute value ${av.name} from ${at.name}
    } explanation {
        The value ${av.name} of attribute ${at.name} is not
        used in any model fragment. Maybe you want to
        include it in this model fragment.
    }
}
}

```

B

Experiment

B.1 The Questionnaire

Computer vaardigheid en affiniteit

Hieronder volgt een korte vragenlijst waarin wordt gevraagd naar uw ervaring en affiniteit met computers en programmatuur (software). Sta niet te lang stil bij het geven van een antwoord. De informatie zal met de nodige zorgvuldigheid worden verwerkt, met waarborging van de privacy van de proefpersoon. Voor deze vragenlijst heeft u maximaal 5 minuten de tijd.

Pp. nr.: _____ Sekse: vrouw / man Datum: _____

Ga door met de vragen op de volgende bladzijde.

1

Figure B.1
The MOBUM-HOMERquestionnaire page 1.

1. Hoe vaak maakt u gebruik van een computer?

1	2	3	4	5
heel weinig	weinig	normaal	veel	heel vaak

2. Hoeveel uur denkt u dat u gemiddeld per week een computer gebruikt?

Uren per week (gemiddeld): _____ uren

3. Voor welke redenen gebruikt u de computer? (kies meerdere opties indien van toepassing)

studie	werk	hobby (div.)	nieuws	verveling
--------	------	--------------	--------	-----------

4. Heeft u ervaring met programmeren?

1	2	3	4	5
heel weinig	weinig	enige	veel	heel veel

6. Heeft u de beschikking over een eigen (privé) computer?

Nee	Ja
-----	----

7. Hoe vindt u het om met een computer te werken?

1	2	3	4	5
erg vervelend	vervelend	geen mening	leuk	heel leuk

8. Geef uw mening: Ik werk liever met een computer dan dat ik TV kijk.

1	2	3	4	5
mee oneens	enigszins mee oneens	neutraal	enigszins mee eens	mee eens

2

Figure B.2
The MOBUM-HOMERquestionnaire page 2.

Attitude vragenlijst 1

Hieronder volgt een vragenlijst waarin uw mening wordt gevraagd over de computer programmatuur die u zojuist heeft gebruikt. In deze vragenlijst wordt er gebruik gemaakt van stellingen. Daarbij kunt u aangeven in hoeverre u het eens of oneens bent met een stelling. Wilt u daartoe het antwoord dat het meest van toepassing is omcirkelen? Sta niet te lang stil bij het geven van een antwoord. De informatie zal met de nodige zorgvuldigheid worden verwerkt, met waarborging van de privacy van de proefpersoon. Voor deze vragenlijst heeft u maximaal 15 minuten de tijd.

Volgorde: u heeft eerst met MOBUM gewerkt en vervolgens met HOMER

Pp. nr.: _____

Sekse: vrouw / man

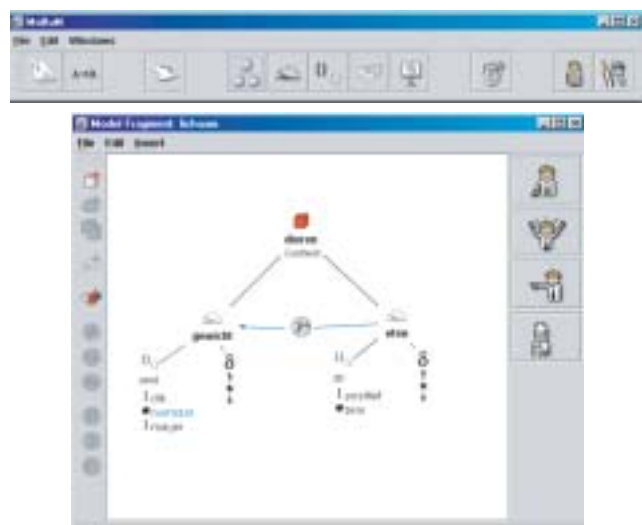
Datum: _____

Ga door naar de volgende bladzijde.

Figure B.3

The MOBUM-HOMERquestionnaire page 3.

Er volgen nu een aantal vragen over **MOBUM**, het programma dat u als eerste heeft gebruikt.



Ga door met de vragen op de volgende bladzijde.

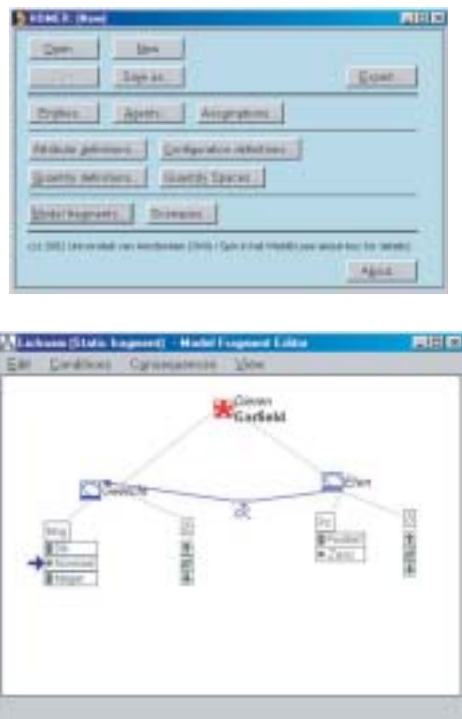
Figure B.4
The MOBUM-HOMERquestionnaire page 4.

1. De diverse schermen van MOBUM zijn helder en duidelijk qua betekenis.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
2. Het programma MOBUM is gemakkelijk te gebruiken.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
3. Het gebruik van kleur in MOBUM is helder en duidelijk qua betekenis.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
4. De foutmeldingen die MOBUM geeft zijn goed.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
5. De 'help' en ondersteuning die MOBUM geeft is goed				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
6. De in MOBUM gebruikte iconen zijn helder en duidelijk qua betekenis.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
7. De navigatie tussen de diverse schermen van MOBUM is gebruikersvriendelijk.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
8. De gedifferentieerde en gepersonifieerde 'help' in MOBUM (zgn Agenten) is aansprekend.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens

5

Figure B.5
The MOBUM-HOMERquestionnaire page 5.

Er volgen nu een aantal vragen over HOMER, het programma dat u als tweede heeft gebruikt.



Ga door met de vragen op de volgende bladzijde.

Figure B.6
The MOBUM-HOMERquestionnaire page 6.

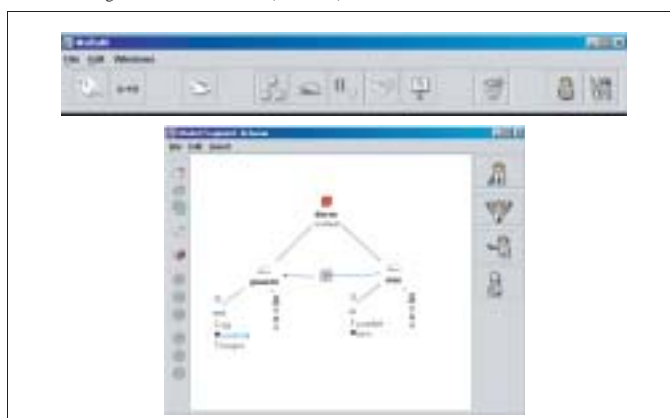
1. De diverse schermen van HOMER zijn helder en duidelijk qua betekenis.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
2. Het programma HOMER is gemakkelijk te gebruiken.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
3. Het gebruik van kleur in HOMER is helder en duidelijk qua betekenis.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
4. De foutmeldingen die HOMER geeft zijn goed.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
5. De 'help' en ondersteuning die HOMER geeft is goed				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
6. De in HOMER gebruikte iconen zijn helder en duidelijk qua betekenis.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
7. De navigatie tussen de diverse schermen van HOMER is gebruikersvriendelijk.				
1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens

7

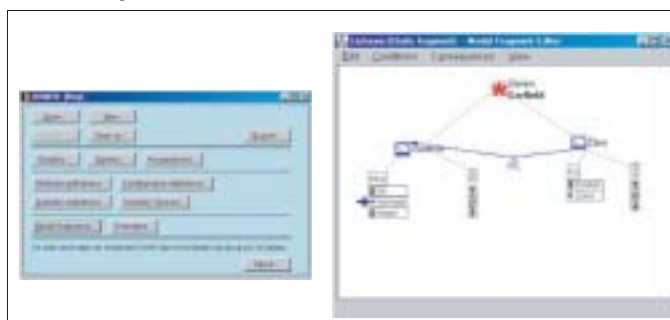
Figure B.7
The MOBUM-HOMERquestionnaire page 7.

Er volgen nu een aantal vragen waarin de programmatuur die u heeft gebruikt onderling worden vergeleken.

U heeft eerst gewerkt met: **MOBUM** (zie onder):



Daarna heeft u gewerkt met: **HOMER** (zie onder):



Ga door met de vragen op de volgende bladzijde.

Figure B.8
The MOBUM-HOMERquestionnaire page 8.

1. De schermen in MOBUM zijn helderder en duidelijker, qua betekenis, dan in HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

2. De MOBUM is gemakkelijker te gebruiken dan HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

3. Het gebruik van kleur in MOBUM is beter dan in HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

4. De foutmeldingen van MOBUM zijn beter dan die van HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

5. De 'help' en ondersteuning van MOBUM is beter dan die van HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

6. De iconen van MOBUM zijn helderder en duidelijker qua betekenis dan die van HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

7. De navigatie tussen de schermen in MOBUM is gemakkelijker dan in HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

8. Alles in overweging nemende vind ik MOBUM een beter programma dan HOMER.

1 mee oneens	2 enigszins mee oneens	3 neutraal	4 enigszins mee eens	5 mee eens
-----------------	------------------------------	---------------	----------------------------	---------------

9

Figure B.9
The MOBUM-HOMERquestionnaire page 9.

B.2 The Experimental Data

B.2.1 The Attitude Questionnaire (A)

Page1

PPN_NR	Computer use	Computer hours per week	Programming experience	Acces to private computer	Attitude towards computer tasks	Computer preference over TV
1.00	3.00	5.00	1.00	1.00	4.00	2.00
2.00	2.00	1.50	1.00	1.00	2.00	1.00
3.00	2.00	2.00	1.00	1.00	2.00	1.00
4.00	4.00	8.00	1.00	1.00	4.00	4.00
5.00	4.00	3.00	1.00	1.00	4.00	3.00
6.00	1.00	1.00	1.00	1.00	4.00	3.00
7.00	4.00	25.00	1.00	1.00	4.00	3.00
8.00	5.00	10.00	2.00	1.00	4.00	3.00
9.00	2.00	2.00	1.00	1.00	3.00	2.00
10.00	3.00	3.00	1.00	1.00	3.00	1.00
11.00	4.00	15.00	2.00	1.00	4.00	3.00
12.00	4.00	5.00	1.00	1.00	4.00	4.00
13.00	3.00	3.00	1.00	1.00	4.00	2.00
14.00	3.00	10.00	2.00	1.00	4.00	4.00
15.00	2.00	2.00	1.00	2.00	4.00	1.00
16.00	3.00	8.00	2.00	1.00	3.00	2.00
17.00	2.00	5.00	1.00	1.00	4.00	1.00
18.00	3.00	2.00	1.00	1.00	3.00	2.00
19.00	2.00	.50	1.00	2.00	3.00	1.00
20.00	3.00	3.00	1.00	1.00	3.00	1.00
21.00	3.00	10.00	1.00	1.00	4.00	2.00
22.00	5.00	30.00	3.00	1.00	3.00	3.00
23.00	3.00	3.50	1.00	1.00	4.50	3.00
24.00	4.00	25.00	2.00	1.00	4.00	5.00
25.00	4.00	19.00	1.00	1.00	4.00	4.00
26.00	4.00	14.00	2.00	1.00	4.00	3.00
27.00	5.00	48.00	3.00	1.00	5.00	5.00
28.00	4.00	10.00	1.00	1.00	4.00	1.00
Grand Total						
Mean	14.50	3.25	9.77	1.36	1.07	3.66
StdDev	8.23	1.04	10.95	.62	.26	.69
Variance	67.67	1.08	119.82	.39	.07	.48

Figure B.10
The results of questionnaire A.

B.2.2 The MOBUM Questionnaire (QM)

Page 1

The order of the tools	PPN_NR	Mobum Question 1	Mobum Question 2	Mobum Question 3	Mobum Question 4	Mobum Question 5	Mobum Question 6	Mobum Question 7
Mobum-Homer	1.00	3.00	3.00	4.00	4.00	4.00	4.00	2.00
Homer-Mobum	2.00	4.00	4.00	3.00	3.00	4.00	5.00	5.00
	3.00	1.00	1.00	3.00	1.00	3.00	3.00	4.00
Mobum-Homer	4.00	5.00	2.00	3.00	3.00	3.00	4.00	4.00
Homer-Mobum	5.00	4.00	4.00	4.00	3.00	4.00	4.00	4.00
	6.00	4.00	4.00	5.00	3.00	5.00	4.00	5.00
Mobum-Homer	7.00	4.00	4.00	4.00	4.00	5.00	3.00	3.00
	8.00	2.00	1.00	1.00	3.00	5.00	3.00	4.00
Homer-Mobum	9.00	3.00	3.00	4.00	4.00	2.00	2.00	3.00
	10.00	4.00	4.00	4.00	3.00	4.00	4.00	4.00
Mobum-Homer	11.00	4.00	2.00	4.00	4.00	3.00	4.00	4.00
	12.00	4.00	2.00	2.00	2.00	4.00	2.00	1.00
Homer-Mobum	13.00	4.00	5.00	5.00	5.00	5.00	5.00	5.00
Mobum-Homer	14.00	5.00	5.00	5.00	4.00	5.00	5.00	5.00
	15.00	4.00	3.00	3.00	3.00	3.00	2.00	4.00
Homer-Mobum	16.00	4.00	4.00	4.00	4.00	5.00	5.00	4.00
Mobum-Homer	17.00	4.00	4.00	5.00	3.00	1.00	4.00	4.00
Homer-Mobum	18.00	2.00	2.00	3.00	4.00	4.00	3.00	3.00
	19.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00
Mobum-Homer	20.00	4.00	2.00	5.00	4.00	5.00	5.00	3.00
	21.00	2.00	2.00	4.00	3.00	2.00	4.00	3.00
Homer-Mobum	22.00	5.00	5.00	4.00	3.00	5.00	4.00	4.00
Mobum-Homer	23.00	4.00	3.00	4.00	2.00	2.00	4.00	5.00
	24.00	1.00	1.00	1.00	3.00	4.00	2.00	2.00
Homer-Mobum	25.00	4.00	4.00	5.00	2.00	1.00	4.00	4.00
	26.00	4.00	4.00	5.00	2.00	2.00	5.00	5.00
	27.00	4.00	5.00	4.00	3.00	3.00	4.00	4.00
Mobum-Homer	28.00	5.00	5.00	4.00	2.00	2.00	5.00	4.00
Grand Total								
Mean	14.50	3.64	3.29	3.79	3.14	3.54	3.82	3.79
StdDev	8.23	1.10	1.30	1.10	.89	1.29	.98	.99
Variance	67.67	1.20	1.69	1.21	.79	1.67	.97	.99

Figure B.11
The results of questionnaire QM.

B.2.3 The HOMER Questionnaire (QH)

Page1

The order of the tools	PPN_NR	Homer Question 1	Homer Question 2	Homer Question 3	Homer Question 4	Homer Question 5	Homer Question 6	Homer Question 7
Mobum-Homer	1.00	3.00	2.00	3.00	4.00	1.00	4.00	4.00
Homer-Mobum	2.00 3.00	2.00 5.00	2.00 2.00	4.00 3.00	3.00 2.00	1.00 1.00	2.00 1.00	4.00 3.00
Mobum-Homer	4.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Homer-Mobum	5.00 6.00	3.00 1.00	3.00 2.00	4.00 4.00	4.00 3.00	3.00 1.00	3.00 2.00	4.00 4.00
Mobum-Homer	7.00 8.00	2.00 1.00	1.00 1.00	3.00 1.00	2.00 1.00	3.00 1.00	2.00 1.00	4.00 2.00
Homer-Mobum	9.00 10.00	2.00 1.00	1.00 1.00	4.00 4.00	2.00 1.00	1.00 1.00	2.00 2.00	2.00 1.00
Mobum-Homer	11.00 12.00	2.00 1.00	4.00 2.00	2.00 4.00	2.00 2.00	1.00 1.00	2.00 2.00	4.00 1.00
Homer-Mobum	13.00	4.00	2.00	4.00	5.00	3.00	4.00	1.00
Mobum-Homer	14.00 15.00	2.00 2.00	3.00 3.00	3.00 2.00	1.00 3.00	1.00 3.00	2.00 3.00	2.00 3.00
Homer-Mobum	16.00	4.00	2.00	4.00	4.00	2.00	4.00	2.00
Mobum-Homer	17.00	1.00	1.00	2.00	1.00	3.00	4.00	1.00
Homer-Mobum	18.00 19.00	1.00 2.00	1.00 2.00	3.00 4.00	2.00 4.00	1.00 2.00	2.00 3.00	2.00 3.00
Mobum-Homer	20.00 21.00	5.00 4.00	4.00 4.00	5.00 4.00	3.00 3.00	2.00 3.00	5.00 4.00	4.00 2.00
Homer-Mobum	22.00	4.00	4.00	3.00	5.00	3.00	4.00	4.00
Mobum-Homer	23.00 24.00	5.00 1.00	2.00 1.00	4.00 1.00	2.00 1.00	3.00 1.00	2.00 2.00	4.00 1.00
Homer-Mobum	25.00 26.00 27.00	2.00 2.00 2.00	3.00 1.00 1.00	2.00 1.00 4.00	3.00 3.00 2.00	3.00 2.00 1.00	4.00 3.00 4.00	2.00 2.00 1.00
Mobum-Homer	28.00	1.00	1.00	1.00	1.00	1.00	1.00	2.00
Grand Total								
Mean	14.50	2.36	2.04	3.00	2.50	1.79	2.68	2.50
StdDev	8.23	1.37	1.07	1.22	1.23	.92	1.16	1.20
Variance	67.67	1.87	1.15	1.48	1.52	.84	1.34	1.44

Figure B.12
The results of questionnaire QH.

B.2.4 The Comparing Questionnaire (Comp)

Page1

PPN_NR	Compare 1	Compare 2	Compare 3	Compare 4	Compare 5	Compare 6	Compare 7	Compare 8
1.00	4.00	3.00	4.00	3.00	5.00	3.00	2.00	3.00
2.00	2.00	2.00	3.00	3.00	1.00	1.00	1.00	1.00
3.00	3.00	3.00	3.00	3.00	2.00	2.00	3.00	3.00
4.00	5.00	5.00	5.00	3.00	5.00	5.00	5.00	5.00
5.00	2.00	2.00	3.00	4.00	2.00	3.00	4.00	3.00
6.00	1.00	1.00	3.00	4.00	1.00	2.00	1.00	1.00
7.00	4.00	4.00	4.00	4.00	5.00	3.00	3.00	5.00
8.00	4.00	5.00	5.00	4.00	5.00	5.00	3.00	4.00
9.00	2.00	4.00	5.00	1.00	3.00	3.00	3.00	1.00
10.00	1.00	1.00	2.00	2.00	1.00	1.00	1.00	1.00
11.00	4.00	2.00	4.00	4.00	4.00	4.00	4.00	4.00
12.00	4.00	3.00	2.00	4.00	5.00	4.00	4.00	4.00
13.00	2.00	1.00	2.00	2.00	2.00	1.00	2.00	1.00
14.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
15.00	4.00	4.00	3.00	4.00	4.00	4.00	3.00	4.00
16.00	1.00	1.00	2.00	2.00	1.00	2.00	2.00	1.00
17.00	3.00	5.00	3.00	5.00	5.00	4.00	5.00	5.00
18.00	4.00	2.00	3.00	1.00	1.00	2.00	3.00	3.00
19.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
20.00	3.00	2.00	3.00	4.00	5.00	5.00	2.00	4.00
21.00	2.00	2.00	4.00	2.00	4.00	4.00	2.00	2.00
22.00	3.00	2.00	3.00	4.00	2.00	3.00	2.00	3.00
23.00	4.00	4.00	4.00	2.00	3.00	4.00	5.00	5.00
24.00	4.00	4.00	3.00	4.00	5.00	3.00	4.00	4.00
25.00	2.00	1.00	2.00	3.00	3.00	1.00	2.00	1.00
26.00	1.00	1.00	2.00	2.00	2.00	1.00	1.00	1.00
27.00	1.00	1.00	2.00	3.00	1.00	2.00	1.00	3.00
28.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
Grand Total								
Mean	14.50	2.93	2.75	3.25	3.18	3.00	2.86	3.00
StdDev	8.23	1.33	1.48	1.08	1.16	1.39	1.38	1.54
Variance	67.67	1.77	2.19	1.16	1.34	2.67	1.90	2.37

Figure B.13
The results of questionnaire Comp.

B.2.5 The Subject’s Evaluation of the Experiment (E)

Page1

Assignment was clear	Software promotes stuctured thinking abou the system	Learning hij building models	Modelling better than study
3.00	3.00	2.00	4.00
4.00	4.00	2.00	4.00
1.00	4.00	4.00	5.00
5.00	4.00	3.00	4.00
4.00	4.00	3.00	4.00
4.00	3.00	4.00	5.00
4.00	4.00	2.00	5.00
4.00	4.00	2.00	2.00
.	.	.	.
2.00	4.00	4.00	5.00
4.00	3.00	2.00	2.00
2.00	3.00	5.00	3.50
4.00	5.00	3.00	5.00
5.00	4.00	2.00	4.00
3.00	2.00	2.00	2.00
4.00	4.00	4.00	2.00
5.00	4.00	5.00	5.00
4.00	4.00	3.00	5.00
4.00	2.00	2.00	2.00
.	.	.	.
3.00	4.00	4.00	4.00
4.00	4.00	4.00	4.00
4.00	3.00	4.00	5.00
4.00	4.00	3.00	4.00
4.00	3.00	4.00	5.00
2.00	4.00	4.00	4.00
5.00	3.00	4.00	2.00
2.00	4.00	3.00	5.00
Grand Total Sum			
94.00	94.00	84.00	101.50
Mean			
3.62	3.62	3.23	3.90
StdDev			
1.06	.70	.99	1.17
> 3			
69.2%	65.4%	46.2%	76.9%
< 3			
19.2%	7.7%	30.8%	23.1%
In 3 to 3			
11.5%	26.9%	23.1%	.0%

Figure B.14
The results of the general evaluation.

B.2.6 The Scored Productivity

Page1

PPN_NR	Homer Total	Mobum Total	Correct scores for mobum	Correct scores for homer
1.00	21.00	24.00	22.00	14.00
2.00	18.00	28.00	7.00	23.00
3.00	17.00	16.00	14.00	22.00
4.00	7.00	24.00	24.00	3.00
5.00	24.00	27.00	4.00	5.00
6.00	17.00	30.00	14.00	18.00
7.00	17.00	27.00	61.00	59.00
8.00	8.00	19.00	14.00	4.00
9.00	14.00	21.00	6.00	14.00
10.00	11.00	27.00	7.00	18.00
11.00	17.00	25.00	11.00	16.00
12.00	13.00	17.00	7.00	8.00
13.00	23.00	34.00	23.00	18.00
14.00	14.00	34.00	58.00	13.00
15.00	19.00	22.00	14.00	5.00
16.00	22.00	30.00	3.00	7.00
17.00	13.00	25.00	52.00	12.00
18.00	12.00	21.00	6.00	5.00
19.00	20.00	28.00	6.00	3.00
20.00	28.00	28.00	18.00	12.00
21.00	24.00	20.00	10.00	18.00
22.00	27.00	30.00	50.00	24.00
23.00	22.00	24.00	17.00	14.00
24.00	8.00	14.00	11.00	3.00
25.00	19.00	24.00	25.00	24.00
26.00	14.00	27.00	24.00	23.00
27.00	15.00	27.00	20.00	16.00
28.00	8.00	27.00	61.00	23.00
Grand Total				
Mean	14.50	16.86	25.00	21.04
StdDev	8.23	5.79	4.94	18.08
Variance	67.67	33.53	24.44	327.00

Figure B.15
The scores of the subjects.

B.3 The Details of the Statistical Analysis

B.3.1 The QM and QH

	N	Minimum	Maximum	Mean	Std. Deviation
Homer Question 1	28	1.00	5.00	2.3571	1.36665
Homer Question 2	28	1.00	4.00	2.0357	1.07090
Homer Question 3	28	1.00	5.00	3.0000	1.21716
Homer Question 4	28	1.00	5.00	2.5000	1.23228
Homer Question 5	28	1.00	3.00	1.7857	.91721
Homer Question 6	28	1.00	5.00	2.6786	1.15642
Homer Question 7	28	1.00	4.00	2.5000	1.20185
Homer Total	28	7.00	28.00	16.8571	5.79089
Valid N (listwise)	28				

	N	Minimum	Maximum	Mean	Std. Deviation
Mobum Question 1	28	1.00	5.00	3.6429	1.09593
Mobum Question 2	28	1.00	5.00	3.2857	1.30120
Mobum Question 3	28	1.00	5.00	3.7857	1.10075
Mobum Question 4	28	1.00	5.00	3.1429	.89087
Mobum Question 5	28	1.00	5.00	3.5357	1.29048
Mobum Question 6	28	2.00	5.00	3.8214	.98333
Mobum Question 7	28	1.00	5.00	3.7857	.99469
Mobum Total	28	14.00	34.00	25.0000	4.94413
Valid N (listwise)	28				

Table B.1
The questions and their metrics for HOMER and MOBUM: the mean of HOMER is 16 and MOBUM 25 on QM and QH.

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
Mobum-Homer	14	7.9286	7.27973	1.94559	3.7254	12.1318	-4.00	20.00
Homer-Mobum	14	8.3571	4.66752	1.24745	5.6622	11.0521	-1.00	16.00
Total	28	8.1429	6.00441	1.13473	5.8146	10.4711	-4.00	20.00

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	1.286	1	1.286	.034	.854
Within Groups	972.143	26	37.390		
Total	973.429	27			

Table B.2

ANOVA analysis displaying the (lack of) effects of the order on the **difference** (QM - QH) in results of QM and QH.

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
Mobum-Homer	14	39.2143	9.25019	2.47222	33.8734	44.5552	22.00	56.00
Homer-Mobum	14	44.5000	8.08370	2.16046	39.8326	49.1674	33.00	57.00
Total	28	41.8571	8.93895	1.68930	38.3910	45.3233	22.00	57.00

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	195.571	1	195.571	2.592	.119
Within Groups	1961.857	26	75.456		
Total	2157.429	27			

Table B.3

ANOVA analysis displaying the (lack of) effects of the order on the **total** (QM + QH) of results of QM and QH.

Order effects

	N	Mean	Std. Deviation	Minimum	Maximum
Homer Question 1	28	2.3571	1.36665	1.00	5.00
Homer Question 2	28	2.0357	1.07090	1.00	4.00
Homer Question 3	28	3.0000	1.21716	1.00	5.00
Homer Question 4	28	2.5000	1.23228	1.00	5.00
Homer Question 5	28	1.7857	.91721	1.00	3.00
Homer Question 6	28	2.6786	1.15642	1.00	5.00
Homer Question 7	28	2.5000	1.20185	1.00	4.00
Mobum Question 1	28	3.6429	1.09593	1.00	5.00
Mobum Question 2	28	3.2857	1.30120	1.00	5.00
Mobum Question 3	28	3.7857	1.10075	1.00	5.00
Mobum Question 4	28	3.1429	.89087	1.00	5.00
Mobum Question 5	28	3.5357	1.29048	1.00	5.00
Mobum Question 6	28	3.8214	.98333	2.00	5.00
Mobum Question 7	28	3.7857	.99469	1.00	5.00

		N	Mean Rank	Sum of Ranks
Mobum Question 1 - Homer Question 1	Negative Ranks	4(a)	10.75	43.00
	Positive Ranks	20(b)	12.85	257.00
	Ties	4(c)		
	Total	28		
Mobum Question 2 - Homer Question 2	Negative Ranks	4(d)	10.88	43.50
	Positive Ranks	20(e)	12.82	256.50
	Ties	4(f)		
	Total	28		
Mobum Question 3 - Homer Question 3	Negative Ranks	2(g)	6.75	13.50
	Positive Ranks	13(h)	8.19	106.50
	Ties	13(i)		
	Total	28		
Mobum Question 4 - Homer Question 4	Negative Ranks	5(j)	5.70	28.50
	Positive Ranks	13(k)	10.96	142.50
	Ties	10(l)		
	Total	28		
Mobum Question 5 - Homer Question 5	Negative Ranks	4(m)	6.75	27.00
	Positive Ranks	22(n)	14.73	324.00
	Ties	2(o)		
	Total	28		
Mobum Question 6 - Homer Question 6	Negative Ranks	1(p)	4.00	4.00
	Positive Ranks	17(q)	9.82	167.00
	Ties	10(r)		
	Total	28		
Mobum Question 7 - Homer Question 7	Negative Ranks	3(s)	9.33	28.00
	Positive Ranks	21(t)	12.95	272.00
	Ties	4(u)		
	Total	28		

	Mobum Question 1 - Homer Question 1	Mobum Question 2 - Homer Question 2	Mobum Question 3 - Homer Question 3	Mobum Question 4 - Homer Question 4	Mobum Question 5 - Homer Question 5	Mobum Question 6 - Homer Question 6	Mobum Question 7 - Homer Question 7
Z	-3.089(a)	-3.078(a)	-2.679(a)	-2.550(a)	-3.814(a)	-3.599(a)	-3.548(a)
Asymp. Sig. (2-tailed)	.002	.002	.007	.011	.000	.000	.000

Table B.4

The non-parametric analysis of the different questions in both conditions.

R E L I A B I L I T Y A N A L Y S I S - S C A L E (A L P H A)						
Covariance Matrix						
	COMP1	COMP2	COMP3	COMP4	COMP5	
COMP1	1.7725					
COMP2	1.5741	2.1944				
COMP3	.9444	1.2130	1.1574			
COMP4	.7169	.7870	.3241	1.3373		
COMP5	1.6429	1.8241	1.0278	1.0780	2.6706	
COMP6	1.3704	1.5556	1.0741	.8889	1.8148	
COMP7	1.4339	1.6296	.8519	.6931	1.4339	
COMP8	1.7407	1.7778	.8889	1.1111	1.8519	
	COMP6	COMP7	COMP8			
COMP6	1.9259					
COMP7	1.3704	1.9048				
COMP8	1.7407	1.7037	2.3704			
Correlation Matrix						
	COMP1	COMP2	COMP3	COMP4	COMP5	
COMP1	1.0000					
COMP2	.7981	1.0000				
COMP3	.6594	.7611	1.0000			
COMP4	.4657	.4594	.2605	1.0000		
COMP5	.7551	.7535	.5846	.5704	1.0000	
COMP6	.7417	.7567	.7194	.5539	.8002	
COMP7	.7804	.7971	.5737	.4343	.6357	
COMP8	.8492	.7795	.5367	.6241	.7360	
	COMP6	COMP7	COMP8			
COMP6	1.0000					
COMP7	.7155	1.0000				
COMP8	.8147	.8018	1.0000			
R E L I A B I L I T Y A N A L Y S I S - S C A L E (A L P H A)						
N of Cases =		28.0				
Item Variances	Mean	Minimum	Maximum	Range	Max/Min	Variance
	1.9167	1.1574	2.6706	1.5132	2.3074	.2549
Item-total Statistics						
	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Squared Multiple Correlation	Alpha if Item Deleted	
COMP1	21.2143	66.8413	.8657	.8242	.9300	
COMP2	21.3929	64.5437	.8706	.8333	.9292	
COMP3	20.8929	73.6548	.6849	.7466	.9421	
COMP4	20.9643	74.9246	.5594	.4673	.9487	
COMP5	20.9643	63.4431	.8200	.7564	.9338	
COMP6	21.1429	65.9048	.8712	.8398	.9293	
COMP7	21.2857	67.3228	.8050	.7411	.9340	
COMP8	21.1429	63.4603	.8818	.8739	.9284	
Analysis of Variance						
Source of Variation	Sum of Sq.	DF	Mean Square	F	Prob.	
Between People	295.1786	27	10.9325			
Within People	124.7500	196	.6365			
Between Measures	5.9286	7	.8469	1.3472	.2303	
Residual	118.8214	189	.6287			
Total	419.9286	223	1.8831			
Grand Mean	3.0179					
Reliability Coefficients 8 items						
Alpha =	.9425	Standardized item alpha = .9416				

Figure B.16

The analysis of the individual questions in the Comp questionnaire.

R E L I A B I L I T Y A N A L Y S I S - S C A L E (A L P H A)						
Covariance Matrix						
	M1	M2	M3	M4	M5	
M1	1.2011					
M2	1.0317	1.6931				
M3	.6614	.9524	1.2116			
M4	.1270	.2169	.2169	.7937		
M5	.0503	.0635	-.2513	.5873	1.6653	
M6	.5635	.7196	.7011	.1376	.0992	
M7	.4392	.6561	.5079	-.0423	-.0661	
	M6	M7				
M6	.9669					
M7	.5899	.9894				
Correlation Matrix						
	M1	M2	M3	M4	M5	
M1	1.0000					
M2	.7235	1.0000				
M3	.5483	.6649	1.0000			
M4	.1301	.1871	.2212	1.0000		
M5	.0355	.0378	-.1769	.5109	1.0000	
M6	.5229	.5624	.6477	.1570	.0782	
M7	.4029	.5069	.4639	-.0478	-.0515	
	M6	M7				
M6	1.0000					
M7	.6032	1.0000				
R E L I A B I L I T Y A N A L Y S I S - S C A L E (A L P H A)						
N of Cases = 28.0						
Item Variances	Mean	Minimum	Maximum	Range	Max/Min	Variance
	1.2173	.7937	1.6931	.8995	2.1333	.1203
Item-total Statistics						
	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Squared Multiple Correlation	Alpha if Item Deleted	
M1	21.3571	17.4974	.6267	.5458	.6980	
M2	21.7143	15.4709	.7113	.6547	.6704	
M3	21.2143	17.6561	.6029	.6544	.7032	
M4	21.8571	21.1640	.3034	.4011	.7619	
M5	21.4643	21.8135	.0801	.4246	.8228	
M6	21.1786	17.8558	.6765	.5833	.6923	
M7	21.2143	19.2857	.4772	.4352	.7314	
Analysis of Variance						
Source of Variation	Sum of Sq.	DF	Mean Square	F	Prob.	
Between People	94.2857	27	3.4921			
Within People	147.7143	168	.8793			
Between Measures	11.9286	6	1.9881	2.3719	.0319	
Residual	135.7857	162	.8382			
Total	242.0000	195	1.2410			
Grand Mean	3.5714					
Reliability Coefficients 7 items						
Alpha =	.7600	Standardized item alpha =	.7674			

Figure B.17

The analysis of the individual questions in the QM questionnaire.

R E L I A B I L I T Y A N A L Y S I S - S C A L E (A L P H A)						
Covariance Matrix						
	H1	H2	H3	H4	H5	
H1	1.8677					
H2	.8016	1.1468				
H3	.8148	.4074	1.4815			
H4	.9259	.6481	.6667	1.5185		
H5	.5608	.3783	.1481	.5556	.8413	
H6	.7116	.5675	.5556	.9074	.5952	
H7	.7037	.6481	.4074	.6296	.2222	
	H6	H7				
H6	1.3373					
H7	.0926	1.4444				
Correlation Matrix						
	H1	H2	H3	H4	H5	
H1	1.0000					
H2	.5477	1.0000				
H3	.4898	.3126	1.0000			
H4	.5498	.4912	.4445	1.0000		
H5	.4474	.3851	.1327	.4915	1.0000	
H6	.4503	.4582	.3947	.6368	.5612	
H7	.4284	.5036	.2785	.4251	.2016	
	H6	H7				
H6	1.0000					
H7	.0666	1.0000				
R E L I A B I L I T Y A N A L Y S I S - S C A L E (A L P H A)						
N of Cases =		28.0				
Item Variances	Mean	Minimum	Maximum	Range	Max/Min	Variance
	1.3768	.8413	1.8677	1.0265	2.2201	.1030
Item-total Statistics						
	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Squared Multiple Correlation	Alpha if Item Deleted	
H1	14.5000	22.6296	.6950	.5028	.7880	
H2	14.8214	25.4854	.6383	.4733	.8002	
H3	13.8571	26.0529	.4829	.3515	.8243	
H4	14.3571	23.3492	.7277	.5835	.7827	
H5	15.0714	27.7725	.5090	.4200	.8199	
H6	14.1786	25.3373	.5892	.6087	.8069	
H7	14.3571	26.6825	.4355	.4418	.8315	
Analysis of Variance						
Source of Variation	Sum of Sq.	DF	Mean Square	F	Prob.	
Between People	129.3469	27	4.7906			
Within People	158.0000	168	.9405			
Between Measures	27.1327	6	4.5221	5.5979	.0000	
Residual	130.8673	162	.8078			
Total	287.3469	195	1.4736			
Grand Mean	2.4082					
Reliability Coefficients 7 items						
Alpha =		.8314	Standardized item alpha =		.8319	

Figure B.18

The analysis of the individual questions in the QH questionnaire.

B.3.2 The attitude questionnaire

	N	Minimum	Maximum	Mean	Std. Deviation	Variance
Computer use	28	1.00	5.00	3.2500	1.04083	1.083
Computer hours per week	28	.50	48.00	9.7679	10.94640	119.824
Programming experience	28	1.00	3.00	1.3571	.62148	.386
Access to private computer	28	1.00	2.00	1.0714	.26227	.069
Attitude towards computer tasks	28	2.00	5.00	3.6607	.69460	.482
Computer preference over TV	28	1.00	5.00	2.5000	1.26198	1.593
Valid N (listwise)	28					

Table B.5

The results of the general attitude towards computers.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	heel weinig	1	3.6	3.6	3.6
	weinig	6	21.4	21.4	25.0
	normaal	9	32.1	32.1	57.1
	veel	9	32.1	32.1	89.3
	heel vaak	3	10.7	10.7	100.0
	Total	28	100.0	100.0	

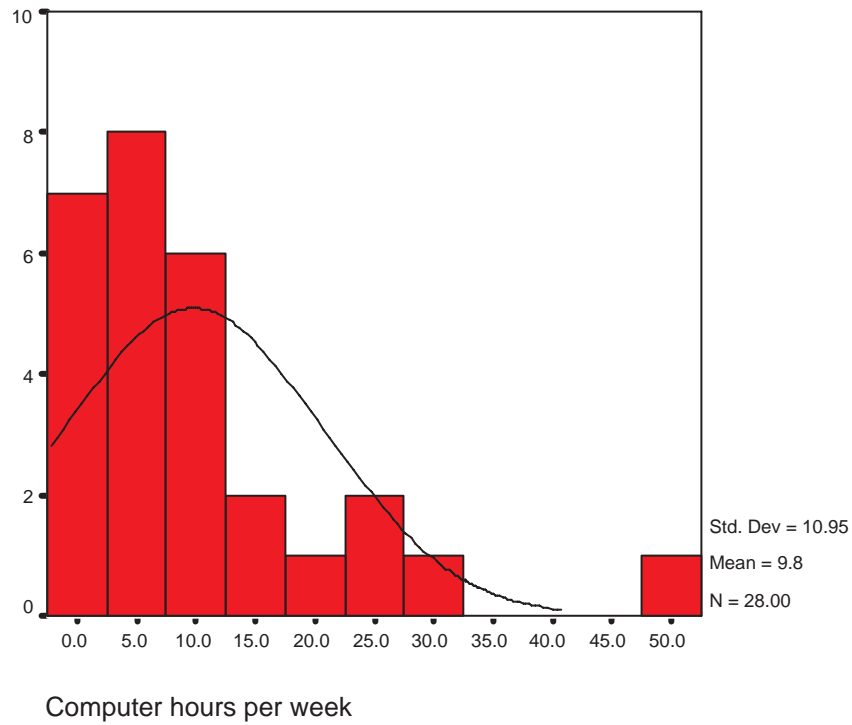
Table B.6

The computer usage of the subjects in the experiment..

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	heel weinig	20	71.4	71.4	71.4
	weinig	6	21.4	21.4	92.9
	normaal	2	7.1	7.1	100.0
	Total	28	100.0	100.0	

Table B.7

The programming experience of the subjects..

**Figure B.19**

The graph displaying hours spend per week.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	ja	26	92.9	92.9	92.9
	nee	2	7.1	7.1	100.0
	Total	28	100.0	100.0	

Table B.8

The access to private computers..

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	vervelend	2	7.1	7.1	7.1
	geen mening	7	25.0	25.0	32.1
	leuk	18	64.3	64.3	96.4
	heel leuk	1	3.6	3.6	100.0
	Total	28	100.0	100.0	

Table B.9

Attitude towards working with a computer.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	mee oneens	8	28.6	28.6	28.6
	enigsins mee oneens	6	21.4	21.4	50.0
	neutraal	8	28.6	28.6	78.6
	enigsins mee eens	4	14.3	14.3	92.9
	mee eens	2	7.1	7.1	100.0
	Total	28	100.0	100.0	

Table B.10
Preference of watching TV compared to working with the computer. 1=preference towards TV, 3=neutral, and 5 is preference towards computer.

B.3.3 The subjects evaluation of the experiment

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	mee oneens	1	3.6	3.8	3.8
	enigsins mee oneens	4	14.3	15.4	19.2
	neutraal	3	10.7	11.5	30.8
	enigsins mee eens	14	50.0	53.8	84.6
	mee eens	4	14.3	15.4	100.0
	Total	26	92.9	100.0	
Missing	.00	2	7.1		
	Total	28	100.0		

Table B.11

The understandability of the assignment.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	enigsins mee oneens	2	7.1	7.7	7.7
	neutraal	7	25.0	26.9	34.6
	enigsins mee eens	16	57.1	61.5	96.2
	mee eens	1	3.6	3.8	100.0
	Total	26	92.9	100.0	
Missing	.00	2	7.1		
	Total	28	100.0		

Table B.12

The software promotes structured thinking about the system.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	enigsins mee oneens	8	28.6	30.8	30.8
	neutraal	6	21.4	23.1	53.8
	enigsins mee eens	10	35.7	38.5	92.3
	mee eens	2	7.1	7.7	100.0
	Total	26	92.9	100.0	
Missing	.00	2	7.1		
	Total	28	100.0		

Table B.13

By building models, people learn how systems work.

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	enigsins mee oneens	6	21.4	23.1	23.1
	3.50	1	3.6	3.8	26.9
	enigsins mee eens	9	32.1	34.6	61.5
	mee eens	10	35.7	38.5	100.0
	Total	26	92.9	100.0	
Missing	.00	2	7.1		
Total		28	100.0		

Table B.14
Interactive learning by a software tool is better that listening or reading about the subject..

List of Figures

2.1	The process of model construction	8
2.2	Qualitative Prediction of Behaviour taken from [23]	15
2.3	A quantity space of water.	16
3.1	The architecture of the entire framework	19
3.2	HOMER displaying a Model Fragment from the Garfield assignment	22
3.3	MOBUM displaying a model fragment from the Garfield example	23
3.4	The swan toolbar and the topology of the different icons.	25
3.5	The SWAN interface with an initial drawing of the U-Tube	26
3.6	The SWAN sketchpad displaying the conditions on which the name is assigned to the object. "Water" is simply put on top of its target. "Flow" is connected with a line to its target. "Container" is grouped with its target	27
3.7	The U-Tube typed. The container is typed as an object, the flow is depicted as a process that takes place in the pipe, and the water in the container is described as a quantity.	28
3.8	The behaviour envisioning of the U-Tube example. a) displays a unbalanced U-Tube with a flow, b) displays the end state, the levels are equal	29
3.9	The causal model pad displaying a simple model. a) The model consisting of two quantities A and B. A has a positive influence on B and A is zero and going up. b) The first state displaying A rising. c) The second state: A is positive and B is rising up. d) The third state: B has become positive	32
4.1	The simulation frame displaying the initial states of the u-tube simulation. The simulation control buttons are located in the toolbar	34
4.2	The basic workings of the force directed algorithm	36
4.3	The different forces on 3 nodes (states) connected by two edges (transitions) in the force directed graph. First there are the springs pulling the states towards each other. Second is the electrical repulsion that pushes all the nodes away from each other.	37
4.4	A picture from GarpApplet displaying the graph of a U-Tube simulation	38
4.5	A screen-shot of one of the most complex models currently modelled in GARP	39
4.6	The U-Tube simulation by the original text-based GARP [14] interface	39
4.7	The table view on the graph on the top, with beneath that a portion of the values of that simulation. This picture is taken from the U-Tube model	40
4.8	The table view showing on of the scenario's of the Cerrado model, one of the most complex simulation currently available	41
4.9	A u-tube graph visualised in a temporal column view	42
4.10	The temporal column view. Each column represents one step in the simulation. This figure displays the same simulation and scenario as figure 4.8 and 4.5 do. State 10 is selected	43
4.11	The causal model inspection tool visualising a causal model	44
4.12	The causal model displaying the U-Tube with all the relations activated	45
4.13	The causal model displaying the U-Tube with all the dependencies activated	46
4.14	The causal model displaying the U-Tube with all the dependencies activated and the entities disabled	47
4.15	The causal model displaying the U-Tube with all the dependencies activated, the entities disabled, and only the P+ dependency enabled	48
4.16	The causal model displaying a highlighted model fragment	48

4.17	The behaviour of the causal model visualisation. a) the entity without any mouse pointer hovering b) the entity highlighted, the mouse pointer has entered the entity c) the mouse pointer has clicked the entity, it has expanded. d) the mouse pointer hovering the quantity, revealing its values and dependencies	49
4.18	The causal model of a state from the cerrado model	50
5.1	HOMER disabling the remove option to prevent the user from removing an essential model part	53
5.2	The HOMER model building environment displaying an error message	54
5.3	A dialog from MOBUM with a disabled OK button. The OK button is disabled because the name field is empty	55
5.4	An error dialog from the MOBUM environment	55
5.5	The Structure Builder in MOBUM displaying the toolbar, a context menu, and a drop down menu	56
5.6	The six avatars acting as personified support	57
5.7	Text provided by the <i>What is?</i> avatar in the Structure Builder	58
5.8	The <i>what is?</i> dialog displaying a help file.	59
5.9	The <i>What can I do next?</i> displaying advice on the contents of the Structure Builder	60
5.10	An model fragment example	61
5.11	The <i>Cross builder help..</i> avatar referring to an object in the SWAN sketchpad	62
5.12	The <i>What can I do next?</i> avatar giving reflections on the Structure Builder	63
5.13	The local avatar dialog advising on a drawing	64
6.1	The means for the total scores on questionnaire QM and QH. The bars represent the 95% conference interval	70
6.2	The means of all four questions of the general evaluation of the experiment. The bars represent the 95% conference interval	74
6.3	The means of the <i>correct</i> scores for HOMER and MOBUM. The bars represent the 95% confidence interval	76
6.4	The means of the correct scores of HOMER and MOBUM by order of the tool	76
A.1	The UML diagram of the basic building blocks	82
A.2	The notation of an isa hierarchy file	83
A.3	The translational architecture overview	84
A.4	An Example of the rule language: two rules in an example rule base	85
A.5	The parsing, translating, and assembling process of the rule base compiler.	87
A.6	Reasoning Engine overview. The note on the left uses the CommonKADS notation, as used in [55]	88
B.1	The MOBUM-HOMERquestionnaire page 1	104
B.2	The MOBUM-HOMERquestionnaire page 2	105
B.3	The MOBUM-HOMERquestionnaire page 3	106
B.4	The MOBUM-HOMERquestionnaire page 4	107
B.5	The MOBUM-HOMERquestionnaire page 5	108
B.6	The MOBUM-HOMERquestionnaire page 6	109
B.7	The MOBUM-HOMERquestionnaire page 7	110
B.8	The MOBUM-HOMERquestionnaire page 8	111
B.9	The MOBUM-HOMERquestionnaire page 9	112
B.10	The results of questionnaire A	113
B.11	The results of questionnaire QM	114
B.12	The results of questionnaire QH	115
B.13	The results of questionnaire Comp	116
B.14	The results of the general evaluation	117
B.15	The scores of the subjects	118
B.16	The analysis of the individual questions in the Comp questionnaire	122
B.17	The analysis of the individual questions in the QM questionnaire	123
B.18	The analysis of the individual questions in the QH questionnaire	124

Experiment	133
B.19 The graph displaying hours spend per week	126

List of Tables

6.1	Sequence of the questionnaires and tools in the experiment	66
6.2	The expectations of the results on QM and QR	69
6.3	The Wilcoxon test (non-parametric) on the difference between the total of HOMER and the total of MOBUM for the QM and QH questionnaires	70
6.4	The Mann-Whitney (non-parametric) test for the <i>first</i> QM and QH questionnaires, that is MOBUM in the MOBUM-HOMER condition and HOMER in the HOMER-MOBUM condition	71
6.5	The Mann-Whitney (non-parametric) test for the <i>second</i> QM and QH questionnaires, that is MOBUM in the MOBUM-HOMER condition and HOMER	71
6.6	The Mann-Whitney of the <i>Comp</i> questionnaire	72
6.7	Summary of the results questionnaire A	73
6.8	The T-Test for comparing the <i>correct</i> of results from MOBUM with the total of results from HOMER	75
B.1	The questions and their metrics for HOMER and MOBUM: the mean of HOMER is 16 and MOBUM 25 on QM and QH	119
B.2	ANOVA analysis displaying the (lack of) effects of the order on the difference (QM - QH) in results of QM and QH	120
B.3	ANOVA analysis displaying the (lack of) effects of the order on the total (QM + QH) of results of QM and QH	120
B.4	The non-parametric analysis of the different questions in both conditions	121
B.5	The results of the general attitude towards computers	125
B.6	The computer usage of the subjects in the experiment.	125
B.7	The programming experience of the subjects.	125
B.8	The access to private computers.	126
B.9	Attitude towards working with a computer	126
B.10	Preference of watching TV compared to working with the computer. 1=preference towards TV, 3=neutral, and 5 is preference towards computer	127
B.11	The understandability of the assignment	128
B.12	The software promotes structured thinking about the system	128
B.13	By building models, people learn how systems work	128
B.14	Interactive learning by a software tool is better than listening or reading about the subject.	129

Index

- abstract syntax tree, 86
- Ainsworth, 7
- Alvarado, 8
- analogical graphics, 9
- arbitrary graphics, 9
- argumentative hypermedia components, 12
- assembling, 87
- attributes, 82
- auditory channel, 6
- automated tutoring, 1
- availability of constraints, 6
- avatars, 65
- bias
 - lookahead, 5
- Bobrow, 13
- breakdown, 51
- bridging representation, 6, 35
- Brna, 6
- catalog component, 12
- causal dependencies, 16
- causal design patterns, 31
- causality, 14
- central executive, 6
- close, 33
- clustering, 10
 - structure-based, 10
- collaborative learning, 1
- common ground, 13
- communicate knowledge, 81
- compiling
 - assembling, 87
 - interpretation, 86
 - parsing, 86
- computational offloading, 5
- computer supported collaborative learning, 79
- condition, 84
- connected mathematics, 4
- connection symbols, 84
- consequence, 85
- construction component, 12
- construction components, 19
- constructionism, 4
- content-based clustering, 10
- context
 - modelling support, 18
- Continuity Rules, 17
- copying, 79
- correct
 - model parts, 68
- correct model parts, 68
- correspondences, 17
- Cox, 6
- CSCL, 79
- design patterns
 - causal, 31
- directed correspondences, 17
- disjunction, 84
- domain-oriented design environments, 12
- dummy model, 29
- Dynamic support, 51
- dynamic support, 69
- effect
 - novelty, 9
- elevation, 10
- erroneous
 - model parts, 68
- erroneous model parts, 68
- executable representation, 86
- executed, 87
- expert advisers, 13
- explanation generation, 1
- exploration
 - incremental, 10
- external representations, 4
- feature, 82
- features, 81
- feedback loop, 27
- Fischer, 12, 13
- focus+context, 10
- Forbus, 7, 13, 14
- force-directed graphs, 35
- frames
 - logical, 10
- full simulation, 33
- ghosting, 10
- graphical constraints, 5

- graphics
 - analogical, 9
 - arbitrary, 9
- grouping, 10
- Heterogeneous inference, 6
- hiding, 10, 44
- highlighting, 44
- horizontal
 - views, 17
- horizontal views, 17
- Identical representations, 6
- incremental exploration, 10
- Inequalities, 16
- Influences, 17
- inherent, 69
- Inherent support, 51
- inherent support, 65
- intelligent tutoring systems, 12
- interactive learning environments, 12
- intermediate
 - modelling support, 18
- intermediate modelling support, 18
- interpretation, 86
- joint commitment, 13
- Jonassen, 3, 4
- Labeke, 6
- Lemon, 6
- library; of model parts, 79
- Lipson, 7
- logical frames, 10
- lookahead, 5
 - mental, 5
- lookahead bias, 5
- mediate, 13
- mental lookahead, 5
- model
 - dummy, 29
- model parts
 - correct, 68
 - erroneous, 68
 - superfluous, 68
- modelling support
 - context, 18
 - intermediate, 18
- modelling support context, 18
- negative influence, 17
- no function in structure, 16
- Norman, 5
- novelty effect, 9
- offloading
 - computational, 5
- ontology, 82
- operations, 5
 - perceptual, 5
- order, 33
- Ordering Rules, 17
- Papert, 4, 12
- parsing, 86
- pasting, 79
- perceptual operations, 5
- Petre, 6
- positive influence, 17
- predicates, 82
- Preece, 11, 12
- Proportionalities, 17
- provide intelligent reflections, 81
- quantitativeness, 14
- re-representation, 5
- re-representing, 5
- reasoning from structure, 15
- representational graphics, 9
- representations
 - external, 4
- Rieber, 9
- rule
 - condition, 84
 - consequence, 85
 - description, 85
 - statement, 84
 - title, 85
 - variables, 84
- semantic fisheye, 44
- semantic zoom, 10
- semantical fisheye, 45
- Shpitalni, 7
- simulated, 1
- simulation mechanisms, 12
- specification component, 12
- specification components, 19
- specificity, 6
- spring layout boxes, 35
- state graph, 15
- statement, 84
- Static support, 51
- static support, 69
- Stenning, 6
- step, 38

- structure-based clustering, 10
- student model, 12
- superfluous
 - model parts, 68
- superfluous model parts, 68
- symbol
 - and, 84
 - connection, 84
 - disjunction, 84
 - or, 84
 - unification, 84
- syntax check, 86
- system of forces, 35

- terminate, 33
- Termination Rules, 17
- tokens, 86
- type, 81, 82

- Ullman, 7
- undirected correspondences, 17
- unification, 84

- views
 - horizontal, 17
- visual containers, 42
- visual-spatial sketchpad, 6

- Wilensky, 4, 13

- Zhang, 4, 5
- zoom
 - semantic, 10
- zooming, 44

Bibliography

- [1] S. E. Ainsworth, P. A. Bibby, and D. J. Wood. Information technology and multiple representations: new opportunities - new problems. *Journal of Information Technology for Teacher Education*, 6(1):93–105, 1997.
- [2] C. Alvarado and R. Davis. Preserving the freedom of paper in a computerbased sketch tool. In *In Proceedings of HCI International 2001*. HCI International, 2001. citeseer.nj.nec.com/alvarado01preserving.html.
- [3] C. J. Alvarado. *A Natural Sketching Environment: Bridging the Computer into Early Stages of Mechanical Design*. PhD thesis, Massachusetts Institute of Technology, May 2000.
- [4] V. Bessa-Machado. Towards understanding model building complexity. 2000.
- [5] V. Bessa-Machado. *Running title: MOBUM (in progress)*. PhD thesis, University of Amsterdam, 2003.
- [6] V. Bessa-Machado and B. Bredeweg. Investigating the model building process with homer. In B. Bredeweg, editor, *Proceedings of the International workshop on Model-based Systems and Qualitative Reasoning for Intelligent Tutoring Systems*, pages 1–13, San Sebastian, Spain, June 2002.
- [7] A.F. Blackwell and R. Hague. Designing a programming language for home automation. In G. Kadoda, editor, *Proceedings of the 13th Annual Workshop of the Psychology of Programming Interest Group (PPIG 2001)*, pages 85–103, 2001.
- [8] D. Bobrow. Dimensions of interaction: A shift of perspective in artificial intelligence, aaai-90 presidential address. *AI Magazine*, 12(3):64–80, 1991.
- [9] A. Bouwer. *Running title: Explanation in Qualitative Simulation*. PhD thesis, University of Amsterdam, 2002. work in progress.
- [10] A. Bouwer, V. Bessa-Machado, and B. Bredeweg. *Interactive Model Building Environments*, chapter The Role of Communication in Learning to Model, pages 155–182. Lawrence Erlbaum Associates, London, UK, 2002.
- [11] A. Bouwer and B. Bredeweg. *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, chapter VisiGarp: Graphical Representation of Qualitative Simulation Models, pages 294–305. IOS-Press/Ohmsha, Japan, Osaka, 2001.
- [12] A. Bouwer and B. Bredeweg. Aggregation of qualitative simulations for explanation. In Angell and J. A. Ortega, editors, *Proceedings of the International workshop on Qualitative Reasoning, QR'02*, pages 11–18, Sitges - Barcelona, Spain, June 10-12 2002.
- [13] U. Brandes and D. Wagner. A bayesian paradigm for dynamic graph layout. In *Proceedings of the Symposium on Graph Drawing GD '97*, Springer-Verlag, pages 236–247, 1997.
- [14] B. Bredeweg. *Approaches to Qualitative Reasoning*. PhD thesis, University of Amsterdam, 1992.
- [15] P. Brna, R. Cox, and J. Good. Learning to think and communicate with diagrams: 14 questions to consider. *Artificial Intelligence Review*, 15(1-2):115–134, March 2001.

- [16] A. L. Brown and J. C. Campione. *Classroom lessons: Integrating cognitive theory and classroom practice*, chapter Guided discovery in a Community of learners, pages 229–270. MIT Press, 1994.
- [17] T. K. Capin, I. Pandzic, N. M. Thalmann, and D. Thalmann. *Virtual Worlds in the Internet*, chapter Realistic Avatars and Autonomous Virtual Humans in VLNET Networked Virtual Environments, pages 157–174. Computer Society Press, 1998.
- [18] R. Cox and P. Brna. Supporting the use of external representations in problem solving: the need for flexible learning environments. *Journal of Artificial Intelligence in Education*, 6(2):239–302, 1995.
- [19] I. Cruz. How to visualize a graph specification and algorithms (part i and ii). [ftp://ftp.cs.brown.edu/pub/papers/compgeo/](http://ftp.cs.brown.edu/pub/papers/compgeo/), January 2001.
- [20] J. de Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24(1-3):7–83, 1984.
- [21] J. H. de Kleer and J. S. Brown. Assumptions and ambiguities in mechanistic mental models. In D. Gentner and A. L. Stevens, editors, *Mental Models*, pages 155–190. Lawrence Erlbaum, Hillsdale, 1983.
- [22] J.H. de Kleer and J.S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.
- [23] K. de Koning. *Model based reasoning about learner behavior*. PhD thesis, University of Amsterdam, 1997.
- [24] A.A. diSessa. *Cognitive Process Instruction*, chapter On 'learnable' representation of knowledge, page 250. The Franklin institute press, 1989.
- [25] R. W. Ferguson and K. D Forbus. Georep: A flexible tool for spatial representation of line drawings. In *Proceedings of the 18th National Conference on Artificial Intelligence*, Austin, Texas, 2000. AAAI Press.
- [26] G. Fischer. Lifelong learning—more than training, 1999.
- [27] K. D. Forbus. Self-explanatory simulators for middle-school science education: a progress repor. In A. Iwasaki, Y.; Farquhar, editor, *Qualitative-Reasoning:-The-Tenth-International-Workshop.*, volume 52. AAAI Press, Menlo Park, CA, USA, 1996.
- [28] K. D. Forbus. Why computer modelling should become a popular hobby. *D-Lib magazine*, October 1996.
- [29] K. D. Forbus. Using qualitative physics to create articulate educational software. *IEEE Expert*, 12(3):32–41, May/June 1997.
- [30] K. D. Forbus, R. W. Ferguson, and J. M. Usher. Towards a computational model of sketching. In *Intelligent User Interfaces*, pages 77–83, 2001.
- [31] K. D. Forbus and J. Usher. Sketching for knowledge caprure: A progress report. January 2002.
- [32] G. W. Furnas. Generalized fisheye views. In *Human Factors in Computing Systems, CHI '86 Conference Proceedings*, pages 16–23, 1986.
- [33] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley Pub Co., 1 edition, January 1995.
- [34] F. Goddijn. Automatische vraaggeneratie bij kwalitative simulatie. Master's thesis, University of Amsterdam, 2002.

- [35] T. Hammond and R. Davis. Tahuti: A geometrical sketch recognition system for uml class diagrams.
- [36] I. Herman, G. Melançon, and M. S. Marshall. Graph visualisation and navigation in information visualisation: A survey. *IEEE Transactions on Visualisation and Computer Graphics*, 6(1):24–43, /2000.
- [37] I. Herman, G. Melancon, and M.S. Marshall. Graph visualisation and navigation in information visualisation. 1998.
- [38] M. L. Huang, P. Eades, and J. Wang. Online animated graph drawing using a modified spring algorithm. *Australian Computer Science Comm.: Proc. 21st Australasian Computer Science Conf., ACSC*, 20(1):17–28, 4–6 1998.
- [39] J. Jellema. Ontwerpen voor ondersteuning - de rol van taakkennis bij ondersteuningsontwerp. Master's thesis, University Of Amsterdam, 2000.
- [40] D. Jonassen. Using cognitive tools to represent problems. Internet, 2002. Submitted to Journal of Research on Technology in Education.
- [41] D. Kimelman, B. Leban, T. Roth, and D. Zernik. Reduction of visual complexity in dynamic graphs. In *Proceedings of the Symposium on Graph Drawing GD '93*, Springer-Verlag, 1994.
- [42] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986.
- [43] J. Larkin and H. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.
- [44] H. Lipson and M. Shpitalni. Conceptual design and analysis by sketching. *Journal of AI in Design and Manufacturing (AIEDAM)*, 14:391–401, 2000.
- [45] J. Nielsen. *Usability Inspection Methods*, chapter Heuristic evaluation, page 30. John Wiley & Sons, New York, NY., 1994.
- [46] D. Norman and J. Zhang. Representations in distributed cognitive tasks. *Cognitive Science*, 18:87–122, 1994.
- [47] S. Papert and I. Harel. *Constructionism*, chapter 1. Ablex Publishing Corporation, 1991.
- [48] M. Petre, A.F. Blackwell, and T.R.G. Green. *Cognitive Questions in Software Visualisation*. MIT Press, 1997.
- [49] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human Computer Interaction*. Addison-Wesley, 1994.
- [50] T. Reichherzer, A. Caas, K. Ford, and P. Hayes. The giant: A classroom collaborator. In *Workshop on Pedagogical Agents of the Fourth International Conference on Intelligent Tutoring Systems (ITS)*, pages 83–86, San Antonio, 1998.
- [51] L. P. Rieber. *Computer Graphics & Learning*. 2000.
- [52] T. Roxborough and A. Sen. Graph clustering using multiway ratio cut. In *Proceedings of the Symposium on Graph Drawing GD '97*, Springer Verlag, pages 291– 196, 1998.
- [53] P. Salles, B. Bredeweg, and R. Winkels. Deriving explanations from qualitative models. In *Proceedings of the World Conference on Artificial Intelligence in Education, AI-ED'97*, pages 474–481, Japan, Osaka, Aug 1997. IOS-Press/Ohmsha.
- [54] M. Scaife and Y. Rogers. External cognition: How do graphical representations work. 1996.

- [55] A. Th. Schreiber, J. M. Akkermans, A. A. Anjewierden, R. de Hoog, N. R. Shadbolt, W. Van de Velde, and B. J. Wielinga. *Engineering and Managing Knowledge, The CommonKADS methodology*. MIT-press, Boston USA, 2000.
- [56] K. Stenning and O. Lemon. Aligning logical and psychological perspectives on diagrammatic reasoning. 15(1-2):29–62, March 2001.
- [57] K. Stenning and J. Oberlander. A cognitive theory of graphical and linguistic reasoning: Logic and implementation. *Cognitive Science*, 19(1):97–140, 1995.
- [58] D. G. Ullman. The importance of drawing in the mechanical design process. *Computer and Graphics*, 14(2):263–274, 1990.
- [59] M. van Hoof. Gkom (tools for knowledge articulation software in education). Master’s thesis, University of Amsterdam, 2003.
- [60] N. van Labeke. Multiple external representations in dynamic geometry: a domain-informed design. In Richard Cox Shaaron Ainsworth, editor, *AI-ED 2001, Workshop Papers, External Representations of AIED: Multiple Forms and Multiple Roles*, pages 24–31, May 2001.
- [61] N. Van Labeke and S.E Ainsworth. Applying the deft framework to the design of multi-representational instructional simulations. In J.D. Moore, C.L. Redfield, and W.L. Johnson, editors, *Proceedings of the 10th International Conference on AI in Education*, pages 314–321, Amsterdam, 2001. IOS Press.
- [62] U. Wilenski. Learning probability through building computational models. In *Proceedings of the Nineteenth International Conference of Mathematics Education*, Recife, Brazil, July 1995.
- [63] A. Woodruff, C. Olston, A. Aiken, M. Chu, V. Ercegovic, M. Lin, M. Spaldings, and M. Stonebraker. Datasplash: A direct manipulation environment for programming semantic zoom visualizations of tabular data. *Journal of Visual Languages and Computing*, 12(1):551–571, July 2001.
- [64] J. Zhang. The nature of external representations in problem solving. *Cognitive Science*, 21:179–217, April-June 1997.