Building Qualitative Models with Homer: A study in Usability and Support

Vania BESSA MACHADO and Bert BREDEWEG

University of Amsterdam
Department of Social Science Informatics
Roetersstraat 15, 1018 WB Amsterdam, The Netherlands
E-mail: {vania,bert}@swi.psy.uva.nl

Abstract

HOMER is a tool that allows learners to create qualitative models of system behaviour. HOMER is organised as a set of builders and tools. Builders capture knowledge and use diagrammatic representations for that purpose. Tools are interactive dialogues for modifying the content of builders. In this paper we present the results of a study examining how learners use HOMER. Two aspects are evaluated, the *usability* of the tool and the *model-building problems* learners may have. The results show that HOMER is usable and that violation of usability factors does not prevent learners from building complex models. Next, to assessing the usability, the goal of the experiment was to investigate the model-building problems that learners have when using tools such as HOMER. These problems are also discussed in this paper. In further research they will be the basis for developing interactive support to improve the usefulness of the tool for learning.

Introduction

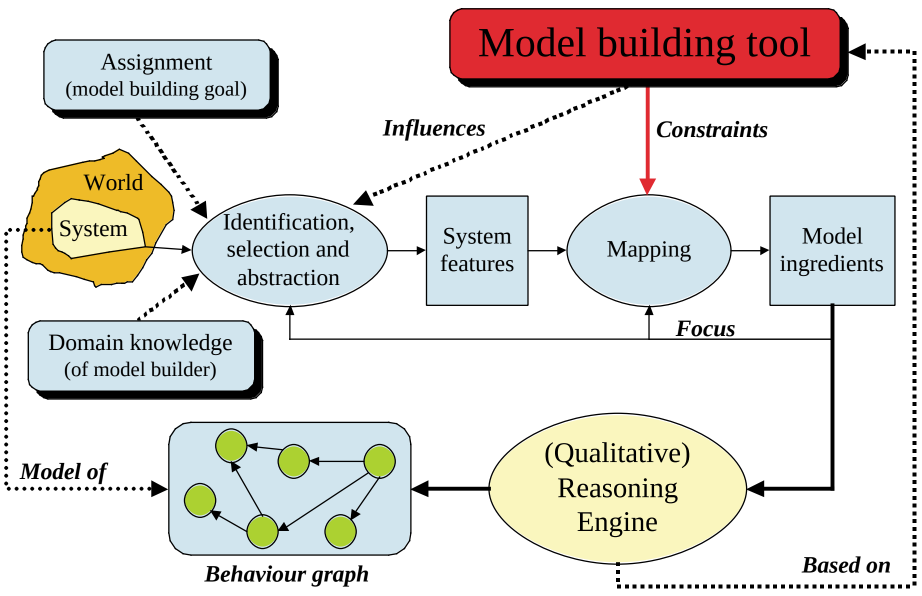
Having learners construct models using graphical notations is an important means to induce learning. Concept maps are well known examples in this respect [12]. Recently, studies have been presented that use a related approach and provide learners with tools to construct diagrams that represent causal explanations of system behaviour, notably 'Betty's Brain' [1] and VMODEL [6]. The primitives provided by these tools for knowledge creation are based on qualitative formalisms, particularly on the Qualitative Process Theory [5].

Research on science teaching has emphasised the importance of learners being able to perform a conceptual (or qualitative) analysis of system behaviour [e g., 10, 4, 12, 7]. Multiple arguments are raised in this respect. One is that experts use such analyses to infer which laws and equations can be applied to a problem situation. Moreover, after deriving a mathematical-based solution a qualitative interpretation is used by these experts to evaluate and explain the solution. Particularly the idea of providing a causal account of the system behaviour is important in this respect. Conceptual analyses are also relevant in situations where mathematical solutions are not available or when we want younger learners to reason about system behaviour who do not yet have the required knowledge of math [8]. In educational settings, qualitative analyses may also facilitate practice with formal representations and consequently foster skills needed for mastering mathematics and programming.

Tools such as 'Betty's Brain' and VMODEL particularly focus on learners assembling causal behaviour diagrams. Our work further develops this idea of 'learning by building models' by having learners construct *full* qualitative models (and run simulations) and thus support learners in a wider range of abilities. We have developed a tool, HOMER, which allows learners to construct qualitative models. The user interface of HOMER consists of a set of builders that use diagrammatic representations for creating knowledge. However, building qualitative models is a complex task [13] and additional support is probably required before learners can effective use HOMER as a tool for learning. In this paper, we present a study that investigates the difficulties that learners encounter when using HOMER. The results are analysed from two perspectives. Problems caused by 'poor usability' [11] of the tool and problems caused by subjects not (fully) understanding how to perform a task. The former can be analysed and repaired in new implementations of the tool. The latter, referred to as *model-building problems*, require augmentation of the tool with online help and other interactive means to support the learner.

Learning by Building Qualitative Models

Building a qualitative model is a complex process during which a multitude of aspects have to be managed by the model builder. At the most general level the problem of building a qualitative model is to create a set of model-fragments (stored in a library) and to specify one or more scenarios. When the simulator is called it uses the model-fragments to predict the behaviour of the system defined in the selected scenario. The output of the simulator (a graph of qualitatively distinct behaviour states) provides feedback on the model construction process. The modelling is successfully completed when for each of the specified scenarios the simulator generates the intended behaviour graph. The idea behind model-fragments is that each fragment represents a general concept relevant to the domain that is being modelled, for instance: a population (ecology), a heat-flow (thermodynamics), or a pressure-area (meteorology). Scenarios are structural descriptions of the particular systems to be reasoned about (see e.g. [3]).


Model building tool
Assignment (model building goal)
Influences
Constraints
World
System
Identification, selection and abstraction
System features
Mapping
Model ingredients
Focus
Domain knowledge (of model builder)
Model of
(Qualitative) Reasoning Engine
Behaviour graph
Based on
Figure 1: Building models and simulations using a tool

But how does a modelling tool relate to this process? We assume that at the highest level it is worthwhile to distinguish between two tasks. One task is concerned with *what* must be captured by the model and possibly includes activities such as identifying, selecting and abstracting the relevant features from a set of systems that exist and manifest behaviour in the real-world. The other task is more concerned with *how* those features should be represented as a coherent set of model ingredients such that the reasoning engine can successfully process them. The latter is referred to as mapping in Figure 1.

Although the two tasks are related, different aspects influence them. In a learning situation the *what* task depends on the assignment given to the learner and on the knowledge the learner has of the domain for which a model has to be constructed. The *how* task, on the other hand, is primarily determined by the representational means the reasoning engine facilitates.

HOMER: a Modelling Tool

HOMER[1] is a tool for constructing qualitative models of system behaviour. Models created with HOMER can be run and inspected using VISIGARP [2]. HOMER consists of builders to create building blocks (entity hierarchy, quantities, quantity-spaces, etc.) and constructs (model-fragments and scenarios). The content of these builders can be manipulated using tools (interactive dialogues). As an example consider the model-fragment builder shown in Figure 2. The model-fragment captures knowledge about an 'Open contained liquid' and holds the entities 'liquid' and 'container'. A configuration defines that the latter 'Contains' the former and an attribute definition specifies that the container is 'Open'. All quantities are assigned to the entity 'Liquid' and have a quantity-space of two values 'zero' and 'plus'. The quantities have 'corresponding' quantity-spaces, which means that they should have the same value from their quantity-spaces (all 'zero' or all 'plus'). Furthermore, the 'Amount' increases the 'Level' and the 'Level' increases the 'Pressure' (specified by dependencies of type proportionality). There is a distinction between *Conditions* and *Consequences* in model-fragments (MF). The former (coloured red in the MF-builder) specifies the conditions under which the later (coloured blue in the MF-builder) are true. The pull-down menu shows the possible manipulation for adding a conditional statement to the model-fragment.

HOMER was designed to prevent learners from making syntactically incorrect models and thus to fully support the *how* task (Figure 1). The user interface is therefore context sensitive and restricts the possible user actions based on (a) the content and (b) the current selections in the builder the learner is working on. As a result, a learner can only perform syntactically correct actions. It may however be the case that a particular action has side effects that the learner is not aware of. For instance, deleting an entry from the entity-hierarchy requires that occurrences of that entity in model-fragments (and scenarios) are also be deleted, in order to preserve the correctness of the model. Notice that this is a recursive feature, because model ingredients connected to that entity (e.g. a quantity) must also be deleted (etc.). HOMER therefore investigates each user action with respect to such side effects, notifies the learner about it, and gives the learner the option to either carry on with the action as planned, or cancel it. As a result a model made in HOMER is by definition always a syntactically correct model.

[1] HOMER is based on VGARP [9]. Visit: http://www.swi.psy.uva.nl/projects/GARP/index.html.

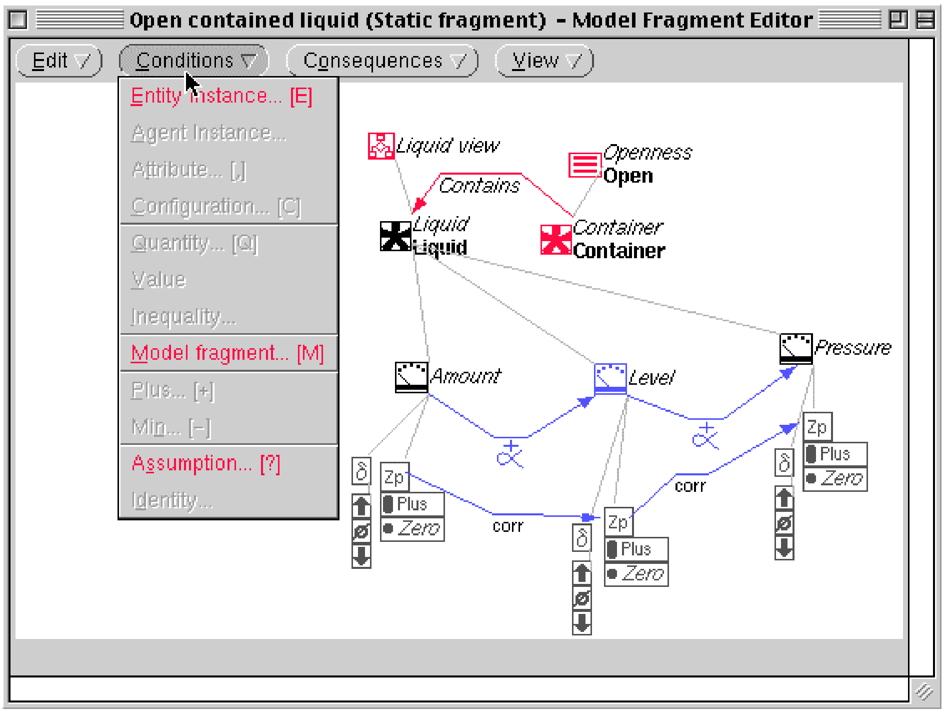


Figure 2: Model-fragment builder showing the content of a specific model-fragment

Method and Subjects

In the experiment the subjects had to construct a simulation model of a U-tube system using HOMER. The subjects received documentation containing the assignment and a short explanation of the screens and icons used in HOMER. Each model-building session was recorded on video, capturing the activity on the computer screen and the verbal expressions uttered by the subject and the experiment leader. Subjects were asked to think aloud as much as possible and thus verbally express what they were doing and the reasons for doing so. The subjects were also encouraged to ask questions during the experiment, because questions are a valuable source of information about the problems encountered. After completing the assignment, the subjects were asked to give a summarising reflection about the bottlenecks they encountered while working with HOMER. This last step was recorded as well. Each session lasted one hour. During the session the subjects could use paper and pencil if they wanted. The subjects were four people from a computer science department. Two of them were researchers and two were master students. All four subjects had experience with artificial intelligence and thus with issues concerning knowledge representation. However, they had not built qualitative models before.

Global Results

Three subjects were able to complete the assignment satisfactorily. They constructed two model-fragments, one for ‘contained liquid’ and one for ‘liquid flow’, one scenario and all the model ingredients needed to actually fill these three constructs. The fourth subject also came far, but did not complete the task of creating a scenario within the available time. Without a scenario it is not possible to run a simulation. From the participants who successfully completed their assignments, two of them actually succeeded in simulating their models using VISIGARP. That is, their models produced behaviour (a graph of qualitatively distinct behaviour states) when simulated. That subjects were able to produce such a result within an hour is encouraging, because the construction of a full qualitative simulation is a complex task. The results of the experiment are analysed from two perspectives. Below we first investigate the overall usability and in the section thereafter we focus on the model-building problems.

Usability of the User Interface

Preferable, we should be able to conclude that HOMER is *usable* for learners and that the problems caused by usability factors do not stand in the way of subjects being able to ‘learn by building models’. In order to assess the usability of the user interface the heuristic evaluation method is used [11] (table 1).

Two of the usability heuristics are not used in our analysis. The principle *Help and documentation* is not examined because the version of HOMER used in the experiment does not have this facility. Recall that the goal of the experiment is to find out what kind of help is required. The heuristics *Match between the (software) system and the real world* is also not examined. One of the goals in our situation is that learners actually *learn* how to use the workbench and by doing so develop a more systematic, partly formal, approach to reasoning about the behaviour of (physical) systems. It is thus most likely that the primitives used in the software are not immediately clear to learners. In fact, our goal is to assess the model-building problems learners have in this respect and use that to further improve tools such as HOMER. Therefore, this heuristic requires a more detailed and also different analysis. This is discussed in the next section.

Visibility of system status
Match between the system and the real world
User control and freedom
Consistency and standards
Error prevention
Recognition rather than recall
Flexibility and efficiency of use
Aesthetic and minimalist design
Help users recognise, diagnose, and recover from errors
Help and documentation

Table 1: Ten Usability Heuristics [11]

A usability problem occurs when a learner sets out to accomplish a task, but fails to do so due to unexpected behaviour of the model-building environment. While working with HOMER, subjects encountered 47 problems that were due to violation of usability heuristics (Figure 2), an average of 12 problems per subject. Most problems related to *Visibility of system status* (27), *Error prevention* (22), and *Consistency and standards* (15).

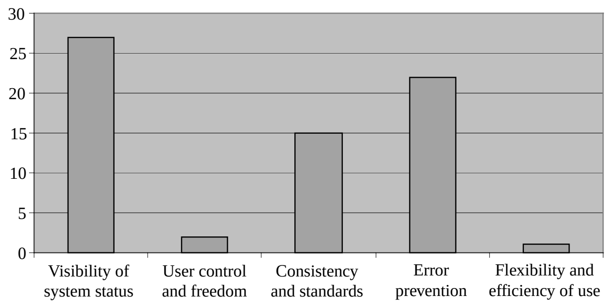


Figure 2: Usability problems due to violation of usability heuristics

Visibility of system status violations were mainly caused by one of factor. When opening a dialogue within builder it displays the latest entries as specified in that dialogue. To change aspects of a model ingredient one can just modify those entries and save them. To define a new model ingredient, the learner should press the NEW button (displayed in each builder) first, that clears the entry fields in the builder. This initially caused confusion by subjects, because the different 'modes' of the builders are not clearly shown. *Error prevention* violations were caused by *default* behaviour of HOMER that was not expected by the subjects. In all builders the current selection changes to the last model ingredient added to that builder. During the design of HOMER this was considered a useful feature because a learner would probably want to further detail the ingredient added last. But in practice it caused confusion. For instance, in the entity builder when the subjects wanted to add a number of subtypes to a *single* entity, somewhat to their surprise they made a chain of subtypes, because each new entity was added as a subtype to the entity created last. Automatically changing the selection, even though clearly shown in the builders, unnecessarily caused the subjects to make errors. A similar problem was caused because HOMER does *not* automatically perform certain default behaviour. For instance, while creating a quantity one can open the quantity-space builder to define a new quantity-space. However, when closing this builder and coming back to the quantity builder, HOMER does not automatically assign the just created quantity-space to the quantity that is being created. During the design of HOMER the idea was that adding a quantity-space by default might cause errors, because learners might not be aware of it being added and end up with a strange model. Defining and adding a quantity-space, so the argument was, are important steps in learning to build models. The tool should therefore enforce learners to think about it, which was realised by *not* automatically performing the task for them. But in practice, the subjects 'thought' they were making a quantity-space for a particular quantity and got confused when HOMER, on closing the quantity builder, complained that there was no quantity-space for that quantity. Finally, *consistency and standards* violations were due to minor inconsistencies in buttons and labels used in different builders for the same functionality.

Some usability heuristics were not violated, and are therefore not shown in Figure 2. One of those heuristics is *Recognition rather than recall*. As the user interface of HOMER is fully graphical, and all user interactions must be initiated using pull-down menus followed by interactive dialogues, all interactions follow this principle and actually do require recognition rather than recall. The heuristic *Help users recognise, diagnose, and recover from errors* is also not violated. As mentioned before, HOMER was designed to prevent users from making syntactically incorrect models and therefore has a context sensitive interface. HOMER also analyses user actions on side effects, notifies the learner about it, and gives the learner the option to either carry on with the action as planned, or cancel it. During the experiment the subjects did not make any complaints on the feedback generated by HOMER in this respect. This can be considered support for the idea that the heuristic is not violated. Finally, the heuristic *Aesthetic and minimalist design* has not been taken into account in our analysis, because we do not consider it of high importance from our research perspective.

As subjects worked longer with HOMER the number of usability problems decreased. For instance, during the task of creating a scenario the subjects encountered much less problems than during the creation of model-fragments. The necessary steps to complete these two tasks are very similar and all subjects created scenarios after they had made the model-fragments. In summary it seems fair to conclude that using HOMER can be learned in a reasonable short time and that the usability of the software was not a significant bottleneck for learners to create simulation models. Still, improvements to the usability of HOMER can and should be made in next versions of the software.

Model-building Problems

A model-building problem was scored when a subject clearly showed an uncertainty concerning the creation of a model ingredient. Often such situations resulted in the subject asking the experiment leader for help. Model-building problems did not always lead to actual errors in the model, particularly not when the subject asked the experiment leader for help. Table 2 summarised the kind of model-building problems encountered by the subjects during the experiment.

Nr. **1**	Type: Entity subtype hierarchy
The awareness of the user that entities should be hierarchically organised.	
Nr. **2**	Type: Defining quantities
Deciding upon the quantities needed to describe the behaviour of a system	
Nr. **3**	Type: Quantity versus quantity-space
Quantity-spaces are modelled as sets of values, independent of specific quantities.	
Nr. **4**	Type: Model-Fragment type
Understanding the role and use of model-fragments.	
Nr. **5**	Type: Structural decomposition
How to structurally decompose the system to be modelled and deciding upon how to relate entities in this respect.	
Nr. **6**	Type: Assigning quantities
Deciding upon which quantity to assign to which entity.	
Nr. **7**	Type: Configuration
Relates to 5, but focussed on the modelling primitive 'configuration' as provided by the tool. Knowing what it means and how to use it.	
Nr. **8**	Type: Attribute versus quantity
Knowing the difference between static (attributes) and dynamic features (quantities).	
Nr. **9**	Type: Defining quantity-spaces
Understanding the rules for defining quantity-spaces.	
Nr. **10**	Type: Dependencies
Understanding of the different types of dependencies and their correct use in model-fragments and scenarios.	
Nr. **11**	Type: Building blocks organisation
Understanding the internal organisation of model-fragments and scenarios. Knowing what type of model ingredients can be placed where.	
Nr. **12**	Type: Generic versus instantiated knowledge
Understanding the overall organisation of creating model ingredients (lists of building blocks), which are then assembled into model-fragments and scenarios.	
Nr. **13**	Type: Specify values
Understanding the role of defining initial values, especially in scenarios.	
Nr. **14**	Type: Quantity-space values
Deciding upon which values to use in a certain quantity-space.	
Nr. **15**	Type: Attribute versus configuration
Knowing the difference between attributes (static features) and configurations (structural relations between entities).	

Table 2: Model-building problems encountered by the subjects

Figure 3 depicts how often each problem type occurred. Notably, most occurrences relate to problem types 10, 11 and 12 (8, 11 and 17 times respectively). *Generic versus instantiated knowledge* problems (type 12) refer to subjects having difficulties in understanding that they were not just building a model of the u-tube, but that they were actually constructing generic model ingredients that might be applicable to a wider range of systems dealing with containers, substances and flows. As a result, they did not always understand that they first had to define generic types (organised sets of entities, quantities, quantity-spaces, configurations, etc.) that could then be used to assemble constructs such as model-fragments and scenarios. Thus, adding a *new* ingredient to a particular model-fragment requires two steps. First, the ingredient must be defined (e.g. a quantity) and then it must be assigned to another ingredient (e.g. an entity). The last step sometimes confused the subjects. Part of this problem can be blamed to violation of the usability heuristic *Error prevention* (e.g. assigning a new created quantity-space to a quantity, see previous section). The other part of the problem can only be solved by the learners acquiring an understanding of what is happening, because it requires deliberate model-building decisions in order to define and connect the ingredients involved. *Building blocks organisation* problems (type 11) refer to subjects making errors, or getting confused, while assembling model-fragments or scenarios. One notable issue was the distinction between *Conditions* and *Consequences* in model-fragments. Subjects had to learn this and while doing so made errors. Related is the fact that certain model ingredients can only be used as *Conditions* (e.g. another MF) or as *Consequences* (e.g. an influence). This is also something the subjects had to learn while developing their modelling skill. Finally, *Dependencies* problems (type 10), which refer to the use of the dependencies as provided by the tool. To become modellers, learners have to acquire knowledge concerning what constraints are available and how they can be used to formulate the conceptual notions that they are developing. A protocol excerpt illustrates a subject's 'mental struggle' in this respect, while he looks at the list of possible dependencies that can be defined:

> *"The flow is directly proportional to the Pressure difference... I guess so... I want to have... proportionality… no... inequality... equal... not really equal but... qualitatively equal I guess..."*

There is no space in this paper to explain all the observed problems in detail. However, some protocol excerpts are worth showing, because they illustrate how subjects are constructing knowledge while building their model. *Defining quantities* (type 2) is an important step in formalising the behaviour of a system. One subject worried about which quantities to use and to which entities they belong:

> *"I wonder if I need to create (entity) 'liquid' and if I need 'Level' to be a property of a container; I don't know if I need 'flow' in this MF... I want to represent (quantity) 'Pressure-Difference' as the difference between the two 'levels'. I am confused. Do I need to create a new quantity?"*

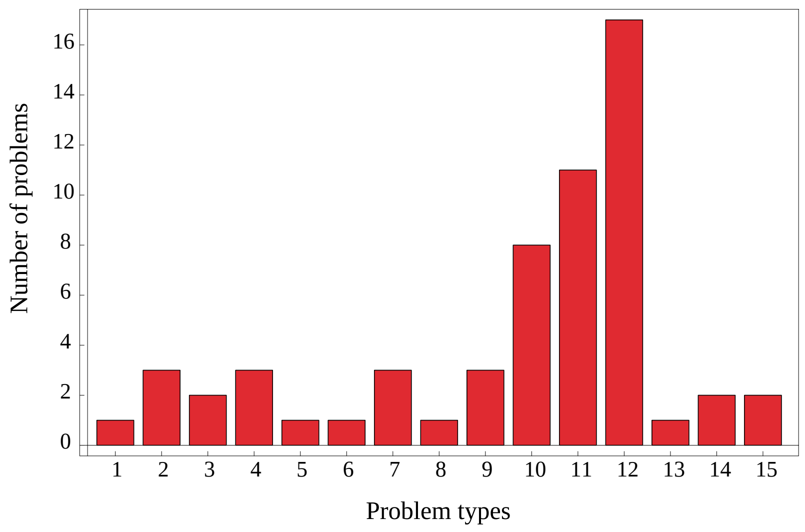


Figure 3: Enumeration of typical model-building problems

Also worth mentioning is *Structural decomposition* (type 5), which refers to issues concerning the structural model of the system at hand. Which entities should be defined and how should they be related? One subject wandered whether to created three occurrences of 'liquid' or only one:

> *"So, that is the U-Tube. (The subject had added three entities, two containers and a pipe, and the structural relation 'connected' between them.) But not the U-Tube with liquid... I am wondering whether I should create a liquid and assign the level of the liquid as a property (left and right container) or whether I should... add two liquids or maybe even three... contained in each container here and here (pointing to all entities on the screen)... It seems a bit silly, I guess... so, I create one liquid."*

Finally, the problem of defining an appropriate set of *Quantity-space values* (type 14). One subject was uncertain whether to give the quantity-space for level (of the liquid column) a maximum value. In end he decided to do so:

> "I think they (the values of the quantity-space) can be zero and plus... I am not sure if I need much more. Let's do it. (The subject creates the point max.)"

The model-building problems from table 2 can be clustered into four groups. Each group captures a typical aspect of the model-building process.

- *Model scope*. Determining which features of the original system to include in the model. It may for instance focus on finding the relevant quantities of a system (e.g. type 2).

- *Model structure.* Determining what to put where in the model. For instance, the issue of deciding on the type and number of model-fragments that are needed (e.g. type 11). The notion of re-use is important in this respect, because it provides guidelines for thinking about how to structure the model. For instance, it is possible to capture all the details of the u-tube system in a single model-fragment. But such a model cannot be used for reasoning about the behaviour of containers, substances and flows in general.

- *Model-building concepts.* Understanding the model-building concepts as provided by the tool. For instance, the difference between attributes and quantities (e.g. type 8), the meaning of an influence (e.g. type 10), or the difference between generic and instance knowledge (e.g. type 12). Model builders need to understand the qualitative ontology as used by the tool. They must learn to use it in order to acquire more advanced insights of the system behaviour they are trying to model.

- *Model representation.* This is related to the 'model-building concepts' category, but now it refers to the actually representation of an idea using the model-building ontology (e.g. type 9). The learner wants to articulate something but does not know *how* to technically formulate that with the options provided by the environment.

Figure 4 shows the total number of problems for each category. Notice that different occurrences of a problem type may fall in different categories. For instance, a problem of type 10 may be a *model concept* (not knowledge the meaning of a dependency) or a *model representation* (not knowing how to *create* a certain dependency). Most problems clearly belong to the category of *model-building concepts*. This means that learners have difficulty in understanding the model-building ontology and that online help should particularly focus on these difficulties. Also notice that this result is exactly what we would hope to find. Tools such as HOMER are meant to have learners acquire a more systematic and formal vocabulary for reasoning about the behaviour of systems. This is precisely what the tool enforces learners to work on. Considering the

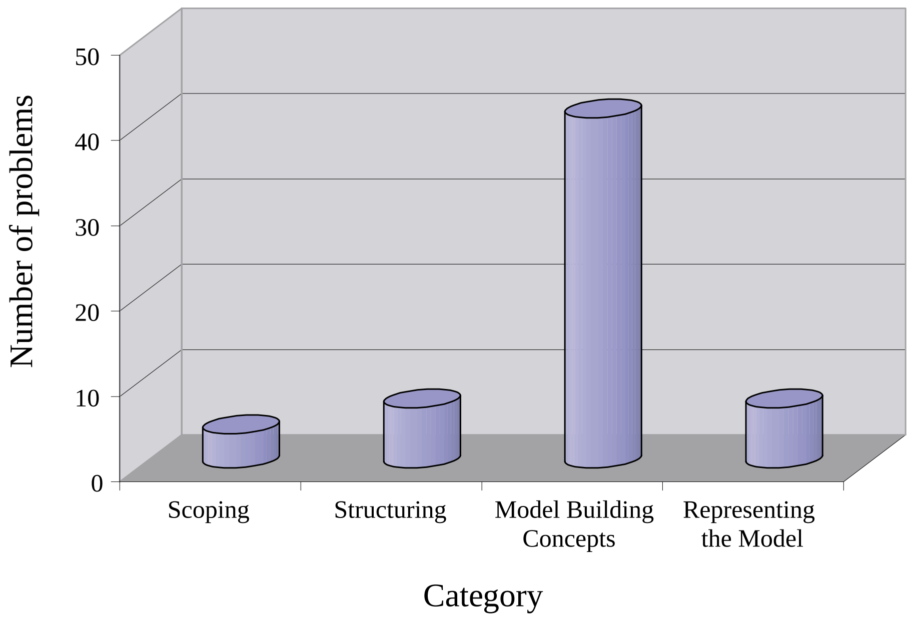


Figure 4: Categories of model-building problems

two tasks initially discussed in section 1, *Model scope* and *Model structure* typically refer to the *what* task whereas *Model representation* refers to the *how* task. *Model-building concepts*, on the other hand, seems to refer to 'how' the modelling tool 'influences' the *what* task (see also Figure 1).

Conclusions and Discussion

Qualitative analysis of system behaviour is an important aspect of science teaching. HOMER is a tool that enables learners to create qualitative models and thereby develop abilities concerning conceptual analysis of system behaviour. However, constructing such models is a difficult task and additional support may be needed in order to have learners effectively use tools such as HOMER. This paper presents a study that analysis HOMER from two perspectives, *usability* and *model-building problems*. Four subjects worked for one hour with the tool constructing a qualitative model. The results show that violation of some usability factors caused difficulties during the modelling process, but they did not prevent subjects from building their models. It seems therefore fair to conclude that the tool is usable. The goal of studying the model-building problems was to investigate how future versions of the tool should be improved. The results suggest that the model-building problems can be clustered into four categories. Most difficulties fall into the *model-building concepts* category. Subjects need support in applying the qualitative ontology as a means to reason more systematic, and formal, about system behaviour. Based on the results gained by the experiment we are developing a set of interactive software agents that support learners in 'doing the right thing' within each builder.

References

[1] Biswas, G., Schwartz, D., Bransford, J. & The Teachable Agents Group at Vanderbilt. (2001) Technology Support for Complex Problem Solving: From SAD Environments to AI. In K. Forbus and P. Feltovich (Eds.). Smart Machines in Education. AAAI Press/MIT Press, Menlo Park California, USA.

[2] Bouwer, A. & Bredeweg, B. (2001) *VisiGarp: Graphical representation of qualitative simulation models.* In J.D. Moore, G. Luckhardt Redfield, and J.L. Johnson (Eds.), Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future, pages 294-305, IOS-Press/Ohmsha, Osaka, Japan.

[3] Bredeweg, B. & Winkels, R. (1998) Qualitative models in interactive learning environments: an introduction. *Interactive Learning Environments*, number 5, volume 1-2, pages 1-18.

[4] Elio, R., & Sharf, P.B. (1990) Modeling novice-to-expert shifts in problem-solving and knowledge organization. *Cognitive Science*, volume 14, pages 579-639.

[5] Forbus, K.D. (1984) Qualitative process theory. *Artificial Intelligence*, volume 24, number 1-3, pages 85–168.

[6] Forbus, K.D, Carney, K., Harris, R. & Sherin, B.L. (2001) A qualitative modeling environment for middle-school students: A progress report. In: G. Biswas (Ed.), The 15th International Workshop on

Qualitative Reasoning, pages 65-72, St. Mary's University, San Antonio, Texas.

[7] Frederiksen, J.R. & White, B.Y. (2002) Conceptualizing and constructing linked models: creating coherence in complex knowledge systems. In P. Brna, M. Baker, K. Stenning and A. Tiberghein (Eds), The role of communication in learning to model, pages 69-96, Lawrence Erlbaum, London.

[8] Jackson, S., Stratford, S.J., Krajcik, J.S. & Soloway, E. (1996) Making system dynamics modeling accessible to pre-college science students. Interactive Learning Environments, volume 4, number 3, pages 233-257.

[9] Jellema, J. (2000) Ontwerpen voor ondersteuning - De rol van taakkennis bij ondersteuningsontwerp. Master thesis, University of Amsterdam, Amsterdam (in Dutch).

[10] Mettes, C.T.C.W. & H. J. Roossink, H.J. (1981) Linking factual and procedural knowledge in solving science problems: A case study in a thermodynamics course. *Instructional Science*, volume 10, pages 333-361.

[11] Nielsen, J. (1994) *Usability Engineering*. Morgan Kaufmann Publishers, San Francisco, USA.

[12] Novak, J.D. & Gowin, D.B. (1984) Learning how to learn. Cambridge University Press, New York, New York.J.D.

[12] Ploetzner, R. & Spada, H. (1998) Constructing quantitative problem representations on the basis of qualitative reasoning, *Interactive Learning Environments*, volume 5, number 1-2, pages 95-108.

[13] Schut, C. & Bredeweg, B. (1996) An overview of approaches to qualitative model construction. *The Knowledge Engineering Review*, 11(1): 1-25.