UvA-DARE (Digital Academic Repository)

Supporting the Construction of Qualitative Knowledge models
Bessa Machado, V.

Citation for published version (APA):
Bessa Machado, V. (2004). Supporting the Construction of Qualitative Knowledge models

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

Download date: 01 Dec 2018
The HOMER Experiment

This chapter presents an experiment carried out to evaluate the design implemented within our first system for building qualitative models (HOMER). The chapter is divided into two main areas. The first four sections deal with the experimental procedure and the presentation of the implemented system. The second part, presents the experiment results and their analysis. The analysis is structured around three main areas, a) validating the task analysis as presented in Chapter 2, b) assessing the tool’s usability in supporting the model building process c) identifying problems and/or misconceptions users encountered when building simulation models with the tool.

3.1 Introduction

This chapter reports on a model building experiment using the fully implemented interactive model building environment HOMER\(^1\) [1] for constructing qualitative models of system behaviour. The design of HOMER is based on the prototype VGARP [63]. HOMER embodies the key features as discussed in Chapter 2 concerning the task analysis, the graphical visualisation, and the organisation of model ingredients. The goal of the experiment was to study the model building behaviour of modellers, particularly to determine the difficulties they have when building models. Based on the obtained results the design of an environment for building qualitative models can be further improved. The latter is discussed in Chapter 5.

In Chapter 2 a distinction was introduced between two tasks relevant to building models, namely the what and the how tasks, see also Figure 2.2. The HOMER environment supports the latter. It prevents the model builder from making models that are syntactically incorrect. Running a qualitative model built with HOMER, using the reasoning engine GARP [16], will always produce a simulation. Being able to use HOMER relieves the model builder from having to be a PROLOG or LISP expert before being able to build a qualitative model. This is an important step towards user-friendly environments

\(^1\)HOMER was designed and implemented as a collaboration between the University of Amsterdam (SWI) and the company "Spin in het Web".
for model building. However, there may be problems that users may have when building models. This chapter investigates those problems.

What problems can model builders using environments such as HOMER have? Firstly, the rational task analysis as presented in Chapter 2 may be incorrect or sub-optimal for actual users. A rational task analysis results in an analysis from a 'logical' point of view: given the model ingredient types and their interrelationships it reflects a kind of optimal approach to building models. But for human model builders this may not be the best approach. Thus we need to determine whether the way human modellers want to build models is supported by the environment. Three questions can be pointed out.

- Is the task structure adequate for modellers?
- Are all required tasks considered useful by modellers?
- Are there tasks that modellers want to perform that are not supported?

A second issue that may cause problems for model builders concerns the overall usability of HOMER. This closely relates to traditional human-computer interaction issues. The question is whether the way the user has to operate the builders and tools in HOMER is easy to use and relates to the way computer software is 'normally' operated. It should be pointed out that this second question has a strong relative component. A user interface may initially appear non-standard but still turn out very convenient after having learned how to use it. Thus, the question to be answered here is not whether HOMER optimally adheres to for example, MicroSoft or Apple user interface standards. Instead, the issue is to determine whether the operation is logical, has no inconsistencies and is relatively easy to use.

The third class of issues concern the actual model building. As mentioned before, HOMER was designed with the goal of supporting the how task. But how does that relate to the what part? Having a fully operational environment such as HOMER allows us to further study the problems users have in this respect. Two questions can be pointed out.

- What problems do modellers have in determining what to include in a specific model?
- What problems do modellers have in actually representing the things they want to capture in a model in terms of the representations provided by HOMER?

To investigate the issues mentioned above a formative evaluation study of HOMER has been performed. Subjects were asked to build models in order to observe their model building behaviour and discover information about their problems and difficulties. The obtained results are presented and discussed in this chapter. The contents of the chapter is as follows. Section 3.2 introduces the U-Tube system. This system is used to explain the HOMER environment, presented in Section 3.3, and it is the object of the model building task given to the subjects in the presented study. Section 3.4 describes the method used to perform the study. Section 3.5 enumerates and discusses the results from the experiment. Finally, Section 3.6 summarises the important insights obtained in the experiment.
3.2 Subject Matter: The U-Tube System

This section introduces the U-Tube system that we use to describe the HOMER environment. Also, building a model of the U-Tube was the task that the participants had to perform in the experiment reported here. As an introduction, consider the U-Tube shown in Figure 3.1. The U-Tube is a system consisting of two containers, filled with liquid, which are connected via a pipe near the bottom of each container. Pressure difference near the bottom of the connected containers determines what will happen. From the container with the higher pressure, liquid will flow to the container with the lower pressure. It is assumed that the pipe has no influence on the behaviour of the system except for facilitating the flow of liquid between the two containers. As a scenario, the level in one container is higher. Running a simulation should thus produce at least two states: the initial disequilibrium and the equilibrium state that succeeds it and for which the levels are equal.

Figure 3.1
An Example U-Tube System.

3.3 HOMER Environment

HOMER [1] is an environment for building qualitative models. Technically, the HOMER environment consists of a set of dedicated builders that embody the task analysis and guidelines for visualisation as presented in Chapter 2. Figure 3.2 shows HOMER’s main window, which is the starting point for interaction.

HOMER consists of seven² builders (Entity Hierarchy, Attribute Definitions, Configurations Definitions, Quantity Definitions, Quantity Spaces, Model Fragment and Scenario Editor), and two additional windows that hold the list of model fragments and scenarios built so far. When the Model Fragments button from the main window is selected, a window that contains the list of created model fragments is displayed. From this window, it is possible to initiate the creation of a new model fragment, delete or edit an existing one. Similar behaviour occurs when the Scenarios button is selected, but then for working on scenarios. In the following, the builders are explained using a model of the U-Tube as running example. Figure 3.3 gives the meaning of the icons employed in Homer.

²In fact nine builders are used in Homer. Next to the seven builders discussed here HOMER also allows the creation of assumption and agent hierarchies. However, these builders are not used in the experiment discussed in this chapter.
CHAPTER 3. THE HOMER EXPERIMENT

Figure 3.2
Homer Main Window.

Figure 3.3
Summary of the icons employed in Homer.

The **Entity Hierarchy** builder supports the definition of the entities that represent the domain. The hierarchical relationships between these entities must be modelled here as well. Figure 3.4 shows an entity’s hierarchy for the U-Tube system. The entities are organised in a tree and they are automatically displayed vertically, horizontally or as a list by selecting the desired choice from the **View menu**. An icon represents the model ingredient type, a text label is the entity’s class-name given by the user and the arrows represent the subtype relationship between entities. Although the hierarchy in the builder contains also intermediate concepts, from a model point of view at least the following entities should be defined for the U-Tube system: container, water and path.

The **Attribute definitions** builder can be used to create the attributes that can belong to entities. The builder organises attributes and its values in lists, Figure 3.5a. By selecting an attribute from the attributes list, its given values are displayed in the possible values list. No icon exists to represent the model-type, instead, the model-type can be inferred from the title of the builder as well as from the list’s title. The user given name to the attribute and its values are represented as text labels. For the U-Tube, the attribute **Openness** has been defined which has open and closed as possible values. It refers to the idea that containers are either open or closed.
The Configuration definitions builder can be used to create the structural relations that can be defined between two entities, Figure 3.5b. A list of configurations is the means of organising them in the builder. No icon is used to represent the model-type and a text label is used for the user given class-name. The model-type is derived from the window title and also from the list browser title. For our example, important structural relations are: connected and contains. When building a model fragment, for instance, this can be used to articulate that a container contains water.

The Quantity definitions builder supports the definition of the generic quantities that may be assigned to an object, Figure 3.6a. The builder consists of the list of created quantities, a list of the allowed quantity spaces for the selected quantity, an area that displays the graphical representation of the selected quantity space, and also a list of all existing quantity spaces. The approach taken in this implementation uses icons for visualising the type of the quantity space values (dots for points and short vertical lines...
for intervals) and organises the values vertically. Text labels are used for the user given class-names for values. Important quantities for the U-Tube model are Amount, Flow, Level and Pressure (at the bottom of the container). In Figure 3.6a, the quantity Amount is selected, which makes the 'Allowed Quantity Spaces' list display the quantity's quantity space Zpm. By selecting the quantity space, its values are displayed in the 'Preview' panel.

![Figure 3.6](image)

Quantity and Quantity Space Builder.

In the Quantity Spaces builder the modeller creates ordered sets of quantity values that quantities may have. These values are a sequence of alternating points and intervals. Similar to the builders described above, it uses a list for displaying the existing quantity spaces. By selecting a quantity space its graphical representation is displayed in the 'Definition' panel, Figure 3.6b. The buttons, next to the 'Definition' panel, are used to define or edit values of a quantity space. Notice that the buttons offer options such as: Add High, Add low, but options to select whether the value is a point or interval are not provided. Only when defining the first value the modeller must choose between point or interval. Thus, the system automatically infers the type, based on the rule that a quantity space must be a sequence of alternating points and intervals.

The Model Fragment Editor allows the modeller to construct the knowledge about the behaviour of entities. This includes the specification of features of entities, such as quantities, the values they have, and the dependencies that exist between the quantities. In this builder, the modeller uses the model ingredients modelled in the previously described builders. Thus, the created building blocks are used to make constructs. Model fragments are 'structures' and following the guidelines for visualising model ingredients, the model ingredients are visualised as a node and, relations to others model ingredients
3.3. HOMER ENVIRONMENT

are shown as arcs. HOMER implements colour coding to visualise the conditions (red) and consequences (blue).

Figures 3.7 and 3.8 show two model fragments of the U-Tube system. The first model fragment, Figure 3.7, specifies the entities Container and Liquid, and relates them by specifying the structural relation Contains, as conditions (coloured red). Furthermore, as consequences, the model fragment specifies Amount and Pressure of a liquid, and relates them to the Level. The Amount is positively proportional to the Level: if the Amount increases then the Level will increase. Likewise, the Level is positively proportional to Pressure: if the Level increases then the Pressure will increase. Additionally, the Amount corresponds to the Level: if Amount is known, then the Level can be derived, and vice versa. For example, if the Level is plus, then the Amount is plus. Finally, Level corresponds directly to the Pressure: if Level is known then the Pressure can be derived, but not vice-versa.

![Figure 3.7](contained_liquid_static_fragment_model_fragment_editor.png)

Model Fragment: Container contains liquid.

The dependencies described in this model fragment are graphically represented as arcs connecting the involved model ingredients. An icon at the middle point of the arc identifies the type of dependency. The implicit relation belongs-to between model ingredients (entity quantity, quantity quantity-space, etc) is graphically represented using a 'light gray' arc.

HOMER also implements context sensitive use of tools to manipulate the content of a builder. For example, in Figure 3.7 as the entity Container is selected only actions that relate to an entity are made available in the menu Conditions: adding an attribute and/or a quantity. Another feature of HOMER is the facility of showing and hiding model ingredients, which can be accessed via the View menu. In the same figure, the graphical representation for derivatives of quantities Amount and Level are hidden while for the quantity pressure it is shown.
Figure 3.8 shows the model fragment for liquid flow. This model fragment requires two model fragments of type 'contained-liquid' to exist. The model fragment then introduces the quantity *Flow* and states that when the *Pressure* at the bottom of a 'contained-liquid' is greater than the *Pressure* at the bottom of the other 'contained-liquid', and there is a path (new added entity as condition), then there will be a *Flow* from one container to the other. The *Flow* rate will be equal to the *Pressure* of one container (high pressure) minus the *Pressure* of the other container. Therefore, the *Flow* has a negative influence on the *Amount* of liquid in the container with higher *Pressure*, and a positive influence on the *Amount* of liquid in the other container: when there is a (positive) flow, the amount of liquid in the first container will decrease and the amount of liquid in the second container will increase.

![Figure 3.8](image)

**Figure 3.8**
Model Fragment Builder: Liquid Flow.

In the **Scenario** builder the modeller defines the situations that can be simulated. Notice that by definition this can only be a 'selection' of the model ingredients defined elsewhere in the model. For instance, there is no point in specifying an entity in a scenario that is not used in any model fragment. Figure 3.9 shows a scenario for the U-Tube which specifies that the *Level* of liquid in the container on the left-hand side is greater than the *Level* of liquid in the container on the right-hand side. Furthermore, the scenario specifies that the *Amount* of water, in both containers, has the value *plus*, which means that some water exists in the containers. In HOMER the graphical representation and the interaction with the model ingredients within a scenario is similar to that for model fragments except that there is no distinction in terms of conditions and consequences.
3.4 Method

In the experiment, the subjects should construct a model of a U-tube system from scratch. The subjects received a document containing the assignment, a short explanation of the qualitative modelling terms employed, as well as a brief introduction to the HOMER Environment (Appendix B).

Each model building session was recorded on video. The camera was pointed at the computer screen with the purpose of capturing the complete sequence of activities performed by the participants and therewith, gathering information about common behaviour of a typical modeller. During the experiment, participants were allowed to ask questions, in fact they were encouraged to do so. This was expected to be a valuable source of information about the problems encountered. The experiment leader could use participant’s questions to elicit further information about a potential problem. Also, during a session participants were asked to think aloud [113], explaining what they were doing and the reasons for doing so. This exposes the modeller’s intentions and can be used to identify their beliefs and misconceptions.

After completing the assignment, the subjects were asked to give a summarising reflection about the bottlenecks and flaws they encountered while working with HOMER as well as suggestions on possible improvements. This last step was recorded as well. Each session lasted approximately one hour. During the session, participants could use paper and pencil to make notes.
Participants

The participants were four people from our department (SWI). Two of them were researchers at this department and the other two were master students. All four participants had experience with Artificial Intelligence and thus with issues concerning knowledge representation and qualitative reasoning. However, they had not built qualitative models before.

3.5 Experiment Results

All participants were able to complete the assignment satisfactorily except for one who did not complete the task of creating a scenario because the official time given to complete the experiment had already passed. From the participants who successfully completed their assignments, two of them actually succeeded in simulating their models using VISIGARP\(^3\) [15]. In this section, the main results of the experiment are summarised and categorised into three topics:

- Task Analysis
- User Interface
- Aspects of Model Building Activities.

In the subsequent analysis, the subjects shall be referred to as P1, P2, P3 and P4.

3.5.1 Task Analysis

To have an impression of how the participants built their models, consider Figure 3.10, which depicts the actions performed by P2.

Sequence of activities

The participant started by creating the entities that he believed to be part of the system to be modelled. Having created the hierarchy of entities, the participant went on with the creation of quantities. While creating a quantity, the participant decided to create the corresponding quantity space. Later on, P2 resumed the building of the quantity from the point where the activity had been interrupted. Actually, in order to create a quantity, a quantity space must also be specified and associated with the quantity. In this situation, the desired quantity space did not exist and the participant decided to initiate the Quantity Space Builder. Following the creation of the quantities, the participant created a static model fragment and started specifying its ingredients. The first step, was *Adding Entities*. Notice that all the entities believed to be part of the model fragment were added first. Only then, the participant went on with selecting the *Add Configuration* activity, with the

\(^3\)Next to the model building environment HOMER, the participants could use VISIGARP for simulating the models they constructed.
help of which P2 created the desired configuration. The participant had not created any configuration yet. It was, therefore necessary to do so at this stage, in order to add it to the
model fragment. Following the specification of the static model fragment, the participant created a process model fragment and started defining it. P2 started with adding the entities, went on with the inclusion of configurations and finally added the quantities. Afterwards, the previously created static model fragment was included as a condition to the process model fragment. Subsequently, an inequality was specified between the included quantities. The participant then proceeded with the creation of a new quantity. As before, while creating this new quantity, called flow, the participant created a new quantity space and finally went back to finish the creation of the quantity. Furthermore, he added the last created quantity to the process model fragment, as well as an influence which used the quantities specified within this model fragment. P2 then created an input system and started specifying its ingredients. The diagram indicates, that the participant kept the same sequence of activities he had previously employed while building the model fragments blocks. P2 first added instances, configurations and quantities and followed this by the specification of an inequality between quantities. The participant concluded the modelling task by setting the initial values of the quantities within the scenario.

The diagrams in Figure 3.11 summarise the activities of all participants. Examining these diagrams some observations can be made. Two of the participants, P1 and P2, went through an identical sequence of activities, namely the creation of entities, quantities, model fragments and a scenario, in that order. The main difference between the sequence of activities performed by these two participants and the steps taken by the remaining two concerns the creation of quantities. While the first two participants had created quantities independently of any context, model fragments or scenario, prior to using them, the other two chose to create them at the moment they were needed. Despite the fact that the activities at the highest level of the HOME R environment can be performed concurrently, see Figure 3.2, three participants initiated the model building process by creating the hierarchy of entities. One participant, P3, did not create a hierarchy of entities. Although not shown in Figure 3.11, P3 gave meaningful names to the entities when including them in the model fragments. By doing so, we may conclude that the participant was aware of the existing entities in the U-Tube system but was not aware that the definition of those entities should have been done in a hierarchical manner and in the dedicated builder.

In summary, examining the diagrams and the protocols, we can conclude that, essentially, the mainstream sequence of activities was: Creation of a hierarchy of entities, followed by the creation of the model fragments and finally the definition of a scenario. Together, these three concepts constitute the important model ingredients in GARP and were also the ones, the subjects focused on most. Other concepts, such as configurations, quantities and dependencies played a more secondary role as they appeared mostly in the context of model fragments and scenarios. When creating the model fragments and the scenario the sequence of the tasks was partially determined by the user interface. For example, a quantity can only be added to the model fragment if an entity to which

---

4As a matter of fact, the reader may notice that the participant made a mistake at this point. P2 had almost completely specified the process model fragment when he realised that the static model fragment, that was already created before, should be included as a condition in this model fragment and, therefore, the entities from the included model fragment should be reused here. This kind of mistake will be further discussed in Section 3.5.3.
that quantity belongs is already added and also selected. However, there still existed a mainstream sequence of activities, which can be summarised as follows:

1. Add Entities

2. Add Attributes/Configurations
3. Add Quantities

4. Add Dependencies

A significant deviation of this sequence occurred when some correction needed to be made. This sequence of activities partially matches the ideal sequence of activities in the task analysis as well as the one suggested by the environment. Looking at Figure 3.2, we can see that, at the highest level, we have the tasks of creating entities, attributes, relations and quantities which match the sequence in which the participants completed the various tasks. However, looking at the diagrams in Figure 3.11, it is clear that the creation of Quantity Spaces was not seen as an independent task by any participant even though the environment suggests that it is an independent task.

**Missing tasks**

Some of the participants pointed out that additional tasks could be added to the interface to make it more complete. However the fact that these suggested tasks were missing did not hamper them in successfully completing their assignments.

- Two of the subjects made drawings on paper before specifying their models. This could be indicative that supporting this task should be part of the environment.

- The subjects pointed out that they missed a complete overview of the model constructed so far. It was mentioned that if they were to model a complex system, it would be harder to keep track of all the details of their models such as the hierarchy of model fragments, the relations between model fragments, and which model fragment is condition to another one.

- An interesting observation was that the subjects were often constructing mental models of the behaviour of their systems, particularly concerning the causal relations between quantities. In fact, they wanted to check if their mental causal relations made sense at all and if they matched the ones specified in their models. This may suggests the need for a tool to construct global causal models.

### 3.5.2 User Interface

In order to assess the usability of the user interface, the heuristic evaluation method was used [83]. Heuristic Evaluation has been widely accepted as a concise test of usability and has been used frequently. The technique rates a user interface by means of a set of usability principles in a cost effective way. In heuristic evaluation evaluators examine the system as in a general evaluation or usage simulation, however the evaluation is guided by a set of heuristics which are concerning key usability issues. The usability heuristics as published by Nielsen are presented in Table 3.1.

The think-aloud protocol, the questions made to the experiment leader, the recorded data as well as the interview at the end of each session were used to register the participants' comments and problems on the user interface at each stage of the model construction process. Hence, the participants were the ones that experienced the problems
3.5. EXPERIMENT RESULTS

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility of system status</td>
<td>The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.</td>
</tr>
<tr>
<td>Match between the system and the real world</td>
<td>The system should speak the user's 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.</td>
</tr>
<tr>
<td>User control and freedom</td>
<td>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.</td>
</tr>
<tr>
<td>Consistency and standards</td>
<td>Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.</td>
</tr>
<tr>
<td>Error prevention</td>
<td>Even better than good error messages is a careful design which prevents a problem from occurring in the first place.</td>
</tr>
<tr>
<td>Recognition rather than recall</td>
<td>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.</td>
</tr>
<tr>
<td>Flexibility and efficiency of use</td>
<td>Accelerators, unseen by the novice user, may often speed up the interaction for the expert user to such an extent that the system can cater for both inexperienced and experienced users. Allow users to tailor frequent actions.</td>
</tr>
<tr>
<td>Aesthetic and minimalist design</td>
<td>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. Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.</td>
</tr>
<tr>
<td>Help users recognize, diagnose, and recover from errors</td>
<td>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 users task, list concrete steps to be carried out, and not be too large.</td>
</tr>
</tbody>
</table>

Table 3.1
Recommended heuristics for interface design.

with the user interface. However, it was the experiment leader and two other HCI experts that examined the encountered problems and judged their compliance with the heuristics afterwards.

Participants Feedback on the User Interface

In this section, we summarise the results of the evaluation by emphasising the most relevant usability issues. Some of the heuristics presented in Table 3.1 are not explored in this evaluation; the reasons for that are the following. The Match between the system and the real world has a different interpretation in this experiment. One of the goals in our situation is that users 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 the users. In fact, our goal is to assess the model building problems users have in this respect and use that to further improve tools such as HOMER. Therefore, this heuris-
tic requires a more detailed and also different analysis. This issue is discussed in Section 3.5.3. The heuristic Help and documentation does not apply because the goal of the experiment was to find out precisely what kind of help is required. Finally the heuristics Recognition rather than recall, Aesthetic and minimalist design, Help users recognise, diagnose, and recover from errors were not violated and therefore they are not mentioned in the results that follow. Consequently, the heuristics that we focus on, are: Visibility of the system’s status, Error Prevention, Consistency and Standards, User Control and Freedom, and Flexibility and Efficiency of Use. We first analyse the most general issues, such as those concerning the overall interface and progressively focus on features specific to individual interface components.

Some of the problems listed below concern the inability of a user in completing a certain task. Other usability problems were detected as a consequence of user complaints and observations. The results are listed following the sequence of the builders in the user interface. However, this does not imply that the problems were encountered in that order. No problems were found with the use of the Quantity Space builder and therefore this builder is not mentioned in the results that follows.

Heuristic Evaluation Analysis

Main window All participants found the ‘New’ Button confusing, three of the four participants tried to initiate building their model by clicking on it (see Figure 3.2). Participants did not seem to notice that they were already in a new model. They expected that it was necessary to press the ‘New’ button in order to initiate the creation of a new session (Violation of: Visibility of the System Status).

Attribute, Configuration, Quantity Space and Quantity Builders Among these four builders with similar look-and-feel only the Configuration Builder does not have a ‘Copy’ button (Violation of: Consistency and Standards).

In the Entity Builder, in order to create a new entity, the user must select the option ‘Add Child’. Attribute, Configuration, Quantity and Quantity Space Builders use the option ‘New’ instead (Violation of: Consistency and Standards).

Entity Builder It was not immediately clear to the participants that they should first select the super type in order to add a new sub-entity to it (Violation of: Visibility of the System Status).

The way HOMER always moves the current selection to the last added entity, caused confusion on behalf of the participants. While creating the entity hierarchy, subjects wanted to add a number of new entities and intuitively expected them to all have the same parent, as they did nothing to change the selection. One of the participants, P4, created four entities and faced this problem twice. Likewise, P1 and P2 created, respectively, four and three entities and faced the problem once. In summary, without exception, all users employing the Entity Builder stumbled on this problem and initially created entities with an incorrect super type (Violation of: Error Prevention).
3.5. EXPERIMENT RESULTS

**Quantity Builder**  None of the participants noticed that in order to create a quantity they should assign a quantity space to it. Initially, all participants just gave a name to their quantities without noticing other options on the screen. This suggest that, the 'Add/Remove' button in the builder was not clearly presented as the next step in the sequence (Violation of: *Visibility of the System status*).

When users create a quantity, they must select a quantity space from the quantity-space list and click on the 'Add/Remove' button to assign the quantity space to the newly created quantity. Even after already having worked with the builder for some time, participants kept forgetting to perform this step. Overall, from the fourteen quantities created by all participants, this error occurred seven times. The error can be explained as follows. When there are no quantity spaces, the user must create a new one from within the Quantity Builder or otherwise abort editing the quantity. This option, 'Edit quantity spaces' caused confusion on behalf of the participants (see Figure 3.12). They expected that by accessing the Quantity Space Builder from within the Quantity Builder, the created quantity space would automatically be assigned to that quantity. But this was not the case. In Homer, the participants are expected to select the created quantity space again in order to associate it with the quantity. All participants faced this problem a number of times (Violation of: *Error Prevention*).

![Figure 3.12](image)

*Figure 3.12*
*Quantity Builder - Quantity Space Builder.*

By default, the last created quantity is always selected for editing. Invariably the participants commented that when they were overwriting the quantities name, they were not sure as to whether they were editing the selected quantity or creating a new one. Some of the participants did not even notice that something was selected and in the hurry of wanting to quickly create a series of quantities, overwrote the one they had previously created, thus losing it. In order to create a new quantity, HOMER expects the user to click
on the New button. This mechanism was found to be counter intuitive by all participants (Violation of: \textit{Visibility of the System status}).

\textbf{Model Fragment Builder} The system allows the user to invoke the other editors via the Model Fragment Builder. For instance, the 'Configuration Definitions’ can be accessed from within the 'Add Configuration’ window, the 'Quantity Definitions’ from within the 'Add Quantity’ window. One of the side effects of this flexibility, was that the participants got lost in the presence of such a high degree of nesting and, as a consequence, did not complete the initially intended task. Indeed, a high degree of concentration is required from the user. For example, three participants, P2, P3 and P4, selected 'Configuration’ from the menu in order to add a configuration. As they had not previously created any configuration, they invoked the 'Configuration Builder’ from within that window. When they returned to the 'Add Configuration’ window, they did not notice that it was still necessary to select the just created configuration. As a result, they closed the window without adding any configuration (Violation of: \textit{Error Prevention}).

Similarly, errors occurred when from the 'Add a new quantity’ menu the participants invoked the 'Quantity Definitions’ (Violation of: \textit{Error Prevention}). In this and in the case above, users must pay attention to what their intentions were in order to avoid incorrect results. Those who did not have problems were the ones that did not use the nested navigation. To illustrate this concern, Figures 3.13 and 3.12 show an allowed sequence of navigation. By selecting 'Quantity...’ from the menu in the Model Fragment Builder (see Figure 3.7), the first window, 'Add a new quantity’, is displayed (Figure 3.13). If the user selects the button 'Edit Definitions...’ in this window, the 'Quantity definitions’ is displayed, Figure 3.12a. In the 'Quantity definitions’, if the user selects the button 'Edit quantity spaces...’, the 'Quantity spaces’ editor is displayed (Figure 3.12b). Getting back to the 'Add a new quantity’, that is closing the other two windows, the users get confused and sometimes cancel the task, instead of saving what they initially set out to do.

As mentioned before, when the participants were using the 'Quantity definitions’ builder just by itself, they did not notice that in the 'Add a new quantity’ window, they also should select the quantity to be added as well as the quantity’s quantity space. In fact, participants committed this error eight times. The worst case was observed with P4 who committed the error four times (Violation of: \textit{Visibility of the System Status}). Concerning the issue of frequently having to switch builders, the participants complained that the graphical differences between the respective windows were not evident and that at some time they did not know anymore which builder they were working on (Violation of: \textit{Visibility of the System Status}).

Finally, it was pointed out that the task of setting a value to a quantity is the only task in which the user does not need to access a dialog box (Violation of: \textit{Consistency and Standards}). Actually, none of the participants succeeded in performing this task without help from the experiment leader.
3.5. EXPERIMENT RESULTS

Overview of User Interface Evaluation

The chart in Figure 3.14 gives an overview of the heuristics violated. The focus of usability problems were on Visibility of System Status, 27, Error Prevention, 22 and Consistency and Standards, 15. However, it was interesting to notice that as subjects worked longer with HOMER the number of usability problems decreased. For instance, during the task of creating a scenario the subjects had significantly fewer problems than during the creation of model fragments. The necessary steps to complete these two tasks are very similar. This indicates that familiarity with the tool did help the participants, as they inevitably had to use the Model Fragment builder before the Scenario builder.

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 the subjects to create simulation models. Still, improvements to the usability of HOMER can and should be made in next versions of the software.

3.5.3 Model building activities

The experiment was effective in detecting flaws, confusions and misconceptions during model building activities. Analysis of the video recordings and moreover, analysis of the transcribed think aloud sessions, permitted the identification of typical problems during the creation of building blocks and model constructs. Table 3.2 presents a summary of the participants’ major problems during the creation of building blocks. Tables 3.3 and 3.4 present the summary corresponding to the creation of model constructs.

Notice that the tables do not represent a sequential order of events. They were organised in this way for the purpose of detecting events common to different individual
participant recordings. In the tables, black circles denote that the participant experienced a problem and could not manage to circumvent it without assistance of the experiment leader. Gray circles denote that the participant had some confusion, misconception or flaw but could solve the problem without help. Hollow circles denote that the participant didn’t face any problems while executing a task. A dash means that the task was not performed.

<table>
<thead>
<tr>
<th>Task</th>
<th>Participants</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create Entities</td>
<td>○</td>
<td>○</td>
<td>-</td>
<td>○</td>
</tr>
<tr>
<td>2</td>
<td>Create Attributes</td>
<td>●</td>
<td>○</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Create Configuration</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>4</td>
<td>Create Quantity</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>6</td>
<td>Create Quantity Space</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

Table 3.2
Summary of building blocks creation.

The following is a discussion of the problems and misconceptions that the participants encountered during the experiment. With the aim of identifying patterns, the problems were classified into fifteen types, after a thorough analysis of the protocols. When the problem or misconception is considered to be an interesting case, a protocol illustrating it is given. For each type of problem, the number of occurrences per participant were registered with the objective of identifying the most frequent types of errors.

**Understanding and using the concept *isa* hierarchy (Type 1).** This category concerns the awareness of the user of the fact that entities should be hierarchically organised. One of the participants, P3, did not create a hierarchy of entities. However, when adding entities to the model constructs, P3 named the entities meaningfully, which indicates that the participant was aware that different entities existed in the U-Tube system. The mis-
3.5. EXPERIMENT RESULTS

<table>
<thead>
<tr>
<th>Participants</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defining MF Super Type</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Add Entities</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Add MF as condition</td>
<td>○</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Add Attributes</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Add Configurations</td>
<td>○</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Add Quantities</td>
<td>●</td>
<td>●</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Specifying Inequalities</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Specifying Causal Dependencies</td>
<td>●</td>
<td>○</td>
<td>●</td>
<td>○</td>
</tr>
</tbody>
</table>

Table 3.3
Summary pertaining to the model fragments creation task.

<table>
<thead>
<tr>
<th>Participants</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Add Entities</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>–</td>
</tr>
<tr>
<td>Add Attributes</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Add Configurations</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>–</td>
</tr>
<tr>
<td>Add Quantities</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>–</td>
</tr>
<tr>
<td>Specifying Inequalities</td>
<td>○</td>
<td>○</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Specifying Inequalities</td>
<td>–</td>
<td>○</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 3.4
Summary associated to the scenario creation task.

description of the participant thus resided in assuming that, by using meaningful names, different entities were being created. During the whole session, P3 did not realise that the entities added to the model fragments (and to the scenario) were all instances of the root of the hierarchy and that thus no entities were created.

Deciding upon which quantities to define (Type 2). One of the principal tasks when modelling a system consists of defining the quantities that describe the behaviour of its entities. Two participants had difficulties with the task of adding quantities to a model fragment. One of the issues concerned the question as to which quantities were relevant in a model fragment. The following protocol gives an example of a participant’s doubt.

P1: 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) ‘PressureDifference’ as the difference between the two ‘levels’. I am confused. Do I need to create a new quantity?

Another issue regards the question as to how such quantities should be defined. Some of the participants needed to ask for help (see protocol below) in order to be able to proceed with the assignment.

P3: How can I say that there is a flow? I mean, if there is difference between the amount of water, we have a flow. Do I need another model fragment to define that flow? ... I need the flow. It is the consequence of this model fragment. But I could also create an attribute saying that there is a flow...
Understanding the relation between a Quantity and its Quantity Space (Type 3). This category concerns the dependency between a quantity and its quantity space. HOMER does not permit the creation of a quantity without a quantity space being assigned to it. When initiating the construction of quantities, a participant, P1 expressed that he wanted to create the quantity height for each container and furthermore to specify that they were equal. However, without specifying a quantity space to be associated with a quantity, this type of dependency cannot be defined. At that stage, the participant had no knowledge about this requirement.

Understanding a Model Fragment Type and its implications (Type 4). This category focusses on the participant's difficulties in understanding the role and structure of model fragments. The following protocol fragment illustrates a participant's misunderstanding. The participant wanted to represent the whole U-Tube with one model fragment. The idea is not completely wrong but the usefulness of model fragments is that the behaviour of the system can be defined in small chunks that can be used in different contexts.

P1: "Could I define the U-Tube as a Static fragment and then... make another fragment to describe the flow of liquid in the U-Tube?"

A most interesting misconception happened when one of the participants intended to represent the flow process in the U-Tube. He had correctly added to the model fragment all the quantities necessary to represent the flow process but still wondered whether to add a dependency or make a new model fragment to represent the flow process. The confusion arose due to the fact that the flow process in the U-Tube has to be represented by dependencies between quantities. A quantity named flow alone does not represent the whole process. At this point, the misconception was explained to the participant in order for him to proceed. Moreover, the participant commented that he wanted to add a model fragment as a consequence of another one. This indicates that the participant was not fully aware of the possibility of reusing a model fragment.

One participant pointed out not to understand the difference between a static and a process model fragment. In this case the experiment leader had to clarify this issue to the participant. Also, P1 was not completely aware of the structure of a model fragment. The participant knew what to model in the model fragment but it wasn't clear to him how to proceed in order to do so. The next protocol fragment illustrates this issue.

P1: I guess I need a process but I don't understand the difference between a static and a process model fragment. A model fragment has no notion of the parts of the model - you cannot say that the U-tube has a container part. You cannot represent this part relation. 'Conditions' - it was not clear to me that the conditions include the entities that exist in the model fragment.

Deciding upon structural decomposition (Type 5). This category is concerned with the way that entities are used in different contexts as opposed to focussing on the entities themselves. In the protocol fragment that follows, the participant had correctly created
the generic entities in the Entity Builder. The problem here was that the participant was confused in how those entities could then be used in the model fragment.

P1: "So, that is the U-Tube. [The participant 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.

Deciding upon which behaviour to assign to which entity (Type 6). One of the participants was not sure as to which entity a quantity should be assigned to. The participant wondered, for instance, if the quantity level belonged to the liquid or to the container entity.

Understanding and using the notion of configuration (Type 7). This category concerns the understanding of the participants of the configuration concept as well as knowing where and how to use it. For instance, after having created a configuration, one of the participants asked where the relation (that configuration) could be specified. It was explained to him that this could only be done in the context of model fragments and scenarios. The participant’s question revealed that a model building concept, in this case a structural relation, had not been clearly understood.

The difference between an Attribute and a Quantity (Type 8). This category deals with understanding the difference between describing an entity’s static features, conceptualised as attributes in the model vocabulary of GARP, and its dynamic behaviour, represented by quantities in the same ontology. One participant intended to use an attribute in order to describe the U-tube’s behaviour. The participant indeed wanted the model to express that the height of one container was bigger than the height of the other container. He questioned whether feature "height" was a static or dynamic feature as it does not change in the specific case of a container. The participant’s argument was interesting. The confusion arose because in the ontology, dependencies such as greater-than, can only be specified between quantities which are dynamic by definition.

Understanding Quantity Space definitions (Type 9). The problems here relate to the rules for defining quantity spaces. For instance, one of the participant had not understood that quantity spaces had to be organised as an alternating series of points and intervals. That same participant wondered whether values from different quantity spaces with the same name were equal or not. As a matter of fact they are not, different quantity space definitions are independent. Another misunderstanding was detected when the participant asked if a quantity space could have negative values. Subsequently he resolved the matter himself and answered the question by saying that the question was meaningless since the
values featured in a quantity space are just labels with no quantitative significance. From these statements, we can deduce that the notion of a quantity space, being an ordered collection of qualitative values, and the reference value zero was not completely understood by the participant.

Understanding and using dependencies (Type 10). This category concerns the understanding of the participants of the different dependencies and their correct use in model fragments and scenarios. The results show that participants often had problems in understanding and using dependencies. For instance, creating an equation relating three quantities, as in \( Q_1 = Q_2 - Q_3 \), proved to be a major hurdle. Participants often confounded the different types of dependencies and how they could be specified. The following protocols illustrate some of the problems encountered:

**P1:** "The flow is directly proportional to the Pressure difference...
I guess so...
I want to have...
proportionality...
no...
inquality...
equal...
not really equal but...
qualitatively equal I guess...

**P2:** We have a \textit{flow} that has a \textit{plus} (i.e. positive) influence...
maybe it would be nice to create another model fragment to define the negative influence...
but is might be good to do it here, since here I have the difference between the one with the most and the one with the least amount of liquid.

**P2:** So I can say that this value [Flow: plus] has a relationship with the amount of liquid.

Understanding the organisation of Model Fragments and Scenarios (Type 11). This item is concerned with the participants' understanding of the different types of knowledge and how they can be specified within model fragments and scenarios.

For instance, one of the participants did not know that derivatives, influences and proportionalities could only be used within model fragments. At a later stage of the model building process, the participant did not know how to start to include knowledge into the model fragment and was puzzled by the existence of the \textit{Conditions} and \textit{Consequences} menu options. Once again, the participant's unfamiliarity with the concepts of "Conditions" and "Consequences" was noted. Furthermore, after having added some correct model ingredients to the model fragment, the participant still demonstrated that he had not totally grasped the concepts.

When initiating the scenario construction, one of the participants wanted to include a model fragment into the scenario. He wanted to reuse the notion of a U-Tube, which had been created using the Model Fragment Builder, inside the scenario.
3.5. EXPERIMENT RESULTS

Understanding the differences between generic knowledge and instantiated knowledge and knowing when to apply either (Type 12). This category concerns the participants’ understanding of the difference between:

- Generic knowledge, which in the GARP ontology includes the entities of the isa-hierarchy, configurations, quantities, quantity spaces and model fragments.

- Instantiated knowledge, categorising instances of generic knowledge used in model scenarios in order to express a specific situation.

The following examples illustrate situations where participants confused the two concepts. When creating a quantity, the user is presented with the list of available generic quantity spaces from which a specific one must be selected and assigned to the quantity. None of the participants seemed to have grasped this concept. Similarly, when adding a quantity to a model fragment, either the quantity and its quantity space must be selected from the sublist of quantity spaces allowed for that quantity since a quantity may have more than one quantity space. One participant had serious difficulties in understanding the 'list' of generic quantity spaces and remarkably committed the error of not selecting it four times.

A participant asked if the names of the model ingredients pertaining to the scenario should match those given to ingredients of the model fragments. This confusion denotes that the participant was not aware of the difference between the knowledge used in model fragments and scenarios. Model fragments represent generic knowledge whereas scenarios contain instantiated knowledge and therefore the names given to the instances are irrelevant as long as their classes are matched with the ones of the corresponding model fragment. Also, the fact that the participant was thinking of the consequences of a scenario, whereas consequences are a term used exclusively in the context of model fragments, proves that the participant was confused about the conceptual differences between a scenario and a model fragment.

Not knowing how to specify values (Type 13). This concerns the participants’ unawareness of the role of defining initial values especially in scenarios. Only two of the participants realised that initial values of quantities should be specified in the scenario in order to have a starting point for the simulation.

Deciding upon which values to include in a quantity space (Type 14). This item refers to the optimal definition of values for expressing the change of behaviour of quantities. One of the participants demonstrated uncertainty as to the definition of values for the quantity spaces.

P3: I think they [the values of the QS] can be zero and plus. I am not sure if I need much more Let's do it.[When creating a quantity space, the participant added another point max]
CHAPTER 3. THE HOMER EXPERIMENT

Attribute-Configuration confusion (Type 15). This category focusses on the conceptual misunderstanding as to the correct use of attributes and configurations. One of the participants created an attribute called connected with the values from and to, whereas he meant a structural relation between two entities. In fact, attributes refer to static features of one and only one entity and, in this case, the participant wanted to establish a relation between two entities, a pipe and a container, in which case the participant should have used a configuration.

![Figure 3.15](image)

**Figure 3.15**
Number of errors classified by type.

Overview of the results on model building activities

The chart shown in Figure 3.15 illustrates the occurrence of errors categorised according to the types mentioned in the previous section. Analysing it, we noticed that errors of types 10, 11 and 12 occur significantly more often than the rest. This came not as surprise because we knew that understanding dependencies (type 10) and model fragments and scenarios (type 11) are in fact difficult concepts to grasp. The occurrence of the enormous number of errors in type 12 may be explained by the fact that the participants did not have the conceptual understanding of reusability. Therefore the participants did not differentiate between the notion of generic knowledge, that could be used in other situations, and instantiated knowledge, which is used for specific situations. Apparently, the participants were only thinking about the given assignment and thus, the conceptual mismatch.

Below, we analyse the various types of faults and try to determine the common nature of the problems which lead to them. The goal of this analysis is to establish the connection between the problems encountered and their causes, relating them to the various stages of the model building process. The model building cycle may be shortly summarised as follows: Firstly, we need to have some knowledge about the system we are going
to model; secondly, we need to organise this knowledge in order to form the model; thirdly, in order to organise the knowledge, we must understand the ontology available for building models; and finally we must master a software program and its respective user interface in order to be able to represent that knowledge using the given ontology in a form which may later be used for simulation.

Looking at the types of errors committed by the participants and taking into account that the modeller must master all four stages of the above-mentioned model building process in order to successfully construct a model, one may cluster the fifteen types of problems mentioned in Section 3.5.3 into four groups. Each group captures a typical aspect of the model building process.

1. **Scoping the Model**: Identification of the knowledge relevant for modelling the system at hand. This category is mainly related to the definition of a canonic set of generic system knowledge elements. For example, which quantities are fundamental to capture the relation between the flow of a liquid and the pressure difference.

2. **Structuring the Model**: Organisation of the model into a working simulation model which can be used to predict the system's behaviour. This step is quite critical. It reflects the participant's view of how to represent the system. In this category the modeller's abstraction capabilities and her/his strive towards promoting reusability plays a central role. For instance, it is always possible to define a model using only one flat model fragment. However, for complex models it proves to be more effective to compose them of several elementary model fragments which may be frequently reused.

3. **Understanding the Model Building Concepts**: Understanding and knowing how to use the ontology for building simulation models. For instance, understanding the difference between attributes and quantities, inequalities and proportionalities, generic and instantiated knowledge.

4. **Representing the Model**: Representing the model of a system model using the available model building tool. This category represents the technical part of the model building process, where the user must get acquainted with some software tool in order to describe her/his ideas in a form which may later be used for simulation. From the protocols we can deduce, for instance, that sometimes the participants knew what they intended to model but they did not find out how to do it using HOMER.

Table 3.5 shows the relation between the types of errors and their relation to the above-mentioned categories. Notice that some types of errors belong to more than one category, which is indicative of the fact that more than one cause may be at a problem's origin. For instance, for errors of Type 10, *Understanding and using dependencies*, two categories may apply. The user may be confusing the notions of influences and proportionalities in which case we are dealing with a problem in understanding a model building concept. But the user may perfectly master the ontology (e.g. \(A = B + C\)) and not know how to express it using the given software environment. In the latter case we have a representation problem.
CHAPTER 3. THE HOMER EXPERIMENT

<table>
<thead>
<tr>
<th>Types</th>
<th>Description</th>
<th>Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Understanding and using the concept &quot;isa&quot;</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>2</td>
<td>Deciding upon which quantities to define</td>
<td>Scoping the Model</td>
</tr>
<tr>
<td>3</td>
<td>Understanding the relation between Quantity and Quantity Space</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>4</td>
<td>Understanding a Model Fragment Type and its implications</td>
<td>Structuring the Model</td>
</tr>
<tr>
<td>5</td>
<td>Deciding upon structural decomposition</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>6</td>
<td>Deciding upon which behavior to assign to which entity</td>
<td>Representing the Model</td>
</tr>
<tr>
<td>7</td>
<td>Understanding and using the notion of configuration</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>8</td>
<td>Attribute and Quantity confusion</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>9</td>
<td>Understanding Quantity Space definition</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>10</td>
<td>Understanding and using dependencies</td>
<td>Representing the Model</td>
</tr>
<tr>
<td>11</td>
<td>Understanding Model Fragment (and Scenario) organization</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>12</td>
<td>Understanding the differences and use of generic (types) versus instance (user given name) knowledge</td>
<td>Structuring the Model</td>
</tr>
<tr>
<td>13</td>
<td>Not knowing how to specify values</td>
<td>Representing the Model</td>
</tr>
<tr>
<td>14</td>
<td>Deciding upon which values to include in a quantity space</td>
<td>Model Building Concept</td>
</tr>
<tr>
<td>15</td>
<td>Attribute-Configuration confusion</td>
<td>Model Building Concept</td>
</tr>
</tbody>
</table>

Table 3.5
Relation between types and categories.

3.6 Summary and Concluding Remarks

In this chapter, we discussed an experiment with a modelling environment for building qualitative models of systems and their behaviour, called HOMER. A short summary of the results is given below.

1. Task Analysis
   The experimental results were satisfactory for validating the task analysis. The implicit task execution sequence of the environment, which is an implementation of the rational task analysis presented in Chapter 2, was generally followed by the users. An exception occurred with the creation of quantity spaces, which, contrary to the task analysis, was not seen as an independent task. A possible explanation for this deviation within HOMER could be the location of the Quantity Space builder button on the main window. Since it is located to the right of the Quantity builder button, the environment intuitively suggests that the quantity space is to be built after the quantity. This is so, because people may think of information flowing from left to right and since the existence of quantity space is a pre-condition to creating a Quantity, the Quantity Space Builder button could better have been placed to the left of the Quantity Builder button.

   The experimental results also indicate that making drawings in order to get a clearer picture of one's model should be a task supported by the environment.
Another interesting aspect was the fact that subjects were constantly constructing mental models of the behaviour of the system. Somehow, this suggests that the environment should support a causal model view (builder), reflecting the state of the model thus far constructed.

2. User Interface
To account for the usability issues of the environment the heuristic evaluation method was applied. The results of Section 3.5.2, show that a number of usability problems were encountered which should be eliminated in a future version. Issues regarding the functionality of the user interface should get special attention. For instance, the participants of the experiment did not like the way in which they were supposed to model a scenario. They suggested that the composition of a scenario should better be done by selecting previously created model ingredients. Still considering functionality, the way in which the interface allows the user to access builders from within builders on the one hand seems to be useful. On the other hand, the environment does not prevent the user from getting lost within the degree of nesting resulting from that flexibility. This flexibility might therefore better not be implemented, as in practice it brings with it more problems than benefits.

3. Model Building Activities
From the results of the experiment a number of problems related to the process of building qualitative models were evident. These problems were categorised into four main classes, according to the stage in which they occurred within the model building cycle:

- scoping the model,
- structuring the model,
- understanding the model building concepts and
representing the model.

Within these four categories, problems caused by failure to understand the model building concepts were remarkably the most frequent ones. This result may be explained as follows: Firstly, this category reflects the understanding of the user of the role of each ontological primitive. These may sometimes be unclear for novices, especially when regarding the use of the different kinds of dependencies. Secondly, the environment forces the user to make a clear distinction between generic and instantiated knowledge. It was frequently noticed that users had difficulties in differentiating the two.

Qualitative analysis of system behaviour is an important aspect of science. HOMER is a tool that enables users 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 is needed in order to have users effectively use tools such as HOMER. The results gained with the experiment are taken into account in the design of a new modelling environment presented in the next chapters.