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
Enabling Support in Building Models

This chapter introduces an approach to support the model building process. It starts with a short review of learning environments that turn out to have important similarities with HOMER. These systems are analysed and compared to HOMER, particularly with respect to the problems encountered by the subjects while constructing their models (as described in Chapter 3). This results in a set of lessons learned: a set of guidelines on how to design a help system as part of an environment for supporting the entire model building process.

4.1 Introduction

As the complexity of software applications increases and computers are being used by all kinds of individuals, the provision of online help\(^1\) is becoming crucial. Computer users expect to be able to use an application with minimal or no training at all, and without having any specialised computer background [68]. These increased user expectations can not be fulfilled by improving only the interface of the systems. In spite of the positive impact on usability that the focus on friendly interfaces may produce, additional help is required [39].

The experiment with HOMER, described in Chapter 3, showed that users of a model building environment need help with accomplishing their tasks, even when they already possess some prior knowledge in building models. In fact, the results of the experiment gave us an indication of the typical problems and misconceptions users have. In this chapter these results are used as input for designing the needed help system for an environment that supports the construction of qualitative models.

We will look at some of the fundamental issues which are relevant when providing help to users. To do so, Section 4.2 describes a set of selected related systems. Section 4.3 presents the lessons learned with the analysed systems relating to the Graphical User Interface (GUI). Techniques currently being employed as well as ongoing research efforts towards the deployment of efficient help systems will be presented in Section 4.4. Section

\(^1\)In this Chapter we consider user support, and use the terms assistance, online help, and help interchangeably.
4.5 discusses how the different forms of feedback given by the selected systems can be incorporated in our design of a support module. Finally, Section 4.7 presents and discusses the approach taken within the scope of this research project.

4.2 Related Educational Software

We now consider different ways in which user support can be provided. A good computer-based support system must aid users in solving problems and at the same time be ready to educate them whenever suitable without letting them drift too far away from their concrete tasks [20]. Without the intention to be exhaustive, we will try to learn from the most successful features in existing support systems in order to design the most adequate one for our model building environment. For that purpose, several software packages, mainly used in educational settings, are analysed. In order to put our analysis into perspective we start by classifying existing educational software systems into four broad categories:

- **Design Environments** Design environments are systems that invariably provide the user with pre-defined building blocks which may be used to build models of real-world systems. A typical example of such a system is CyclePad [46], an environment for designing thermodynamics cycles.

- **Predefined simulations** These systems have in common that simulations are not created by the learners. Learners can therefore only interact with a pre-defined simulation. An example of such a system is SimQuest [72], an authoring system for creating simulation-based learning environments [33].

- **Intelligent Tutoring Systems** Intelligent tutoring systems (ITSs) aim at providing students with individualised feedback based on an analysis of the user's behaviour. Often AI techniques are used to provide such feedback. The Star-Light prototype system [36] is an example of an ITS.

- **Software Agents** Systems based on software agents usually provide personalised support. However, in the present context, our main interest in considering agents is due to their way of interacting with users, e.g. by means of personified and animated characters. Adele [64] is an example of a software agent that includes a personified and animated character.

Specific applications may belong to more than one of these categories. The purpose of the above classification is to help us focus on specific aspects of user support in order to understand in what ways particular features could be considered in the design of our own support system. In what follows, we further explore the categories defined above.

4.2.1 Design environments

Design environments provide pre-defined building blocks and as such offer users, professionals and students alike, the opportunity to practice science/engineering while at the
4.2. RELATED EDUCATIONAL SOFTWARE

same time being valuable low-cost tools for learning in safe environments. Design activities are cognitively demanding and the provision of support in such environments may grant students a smoother access to the complex world of computer-aided design. There are several examples of software that is useful for design work. These include: CAD systems that are now prevalent and indispensable to professional engineers; simulation packages such as MatLab and SimuLink that allow engineers to build and simulate dynamic system models; interactive games such as SimCity that allow users to explore the workings of virtual worlds they built themselves; and model building software such as STELLA [79], Model-It [62], and CyclePad [46]. Of these last three, the former two allow pre-college students to build models of natural phenomena while the latter teaches thermodynamic principles by allowing students to build models of thermodynamic cycles. Each one of the above-mentioned design environments has proven its value in making design activities more accessible to its users.

We have selected STELLA and CyclePad for further analysis as their application and target users are the ones most closely related to HOMER.

STELLA

STELLA is a computer simulation program which provides a framework and a graphical interface for model construction and simulation using the system dynamics methodology [47]. Examples of dynamic systems modelled with STELLA are processes such as growth, death, infiltration, evaporation, etc. The STELLA environment consists of three windows called levels: the Map/Model level, the Interface level and the Equation level. Models are created in the Map/Model level by means of an icon-based language. The Interface level can be set up to provide an easy way to manipulate the options and parameters of the model. At the Equation level one can access the equations automatically generated by the system while the user was building the model. Normally the user does not need to work at this level because STELLA translates the graphical representation of the model created by the user into an underlying set of mathematical equations and thus eases the efforts of the user. STELLA provides multiple representations of the simulation results, such as diagrammatic animations when the simulation is running, charts over time, tables of numeric values and numeric displays thereby facilitating considerably the interpretation of the numerical data.

The icon-based language consists of four building blocks, namely stocks, flows, converters and connectors. Stocks, represented by a rectangular icon, stand for stuff that accumulates. Both physical and non-physical objects may be modelled as stock. Flows, represented by a thick arrow with an attached circle, stand for actions. They fill and drain stocks. Examples of stocks are: water in a cloud, body weight, and anger. The associated flows may respectively be: evaporating/precipitating, gaining/losing, and building/venting. A flow can be connected between two stocks, and a stock can be connected to one or more flows. A circular causal connection that begins at a certain stock, runs through flows to other stocks and then returns to the same stock is called a 'feedback loop', thus a change in one stock affects another stock over time, which in turn affects the original stock. This brings us to the function of the remaining two modelling primitives,
converters, represented by circles, and connectors, represented by thin arrows. Converters either affect the rate of flow or the contents of another converter. Connectors indicate that changes in one thing affect the changes in another thing. Connectors never point to a stock; only flows change stocks. Consider a simplistic model of the population growth as depicted in Figure 4.1. The Birth Rate, a converter, is directly related to the size of the population, a stock, and so is the Death Rate. Thus, information about the current size of the population (the arrows represent connectors) must be conveyed to the rates of births and deaths. Whether the population level tends to rise or fall depends on the relative values of the Birth and Death rates.

**Figure 4.1**
Simple Stella Model of the Population Growth.

The population, a stock, and so is the Death Rate. Thus, information about the current size of the population (the arrows represent connectors) must be conveyed to the rates of births and deaths. Whether the population level tends to rise or fall depends on the relative values of the Birth and Death rates.

**CyclePad**

CyclePad attempts to help engineering undergraduates to learn the important principles of thermodynamics. It provides a conceptual CAD environment where students can design and analyse power plants, refrigerators, and other systems involving thermodynamic cycles. The system is based on existing AI techniques. It uses: "compositional modelling to represent and reason with modelling assumptions, qualitative representations to express the intuitive knowledge of physics needed to detect impossible designs, truth maintenance to provide the basis for explanations, and constraint reasoning and propagation to provide efficient mathematical reasoning" [46].

CyclePad provides a predefined library of components represented by a set of icons (compressors, turbines, pumps, heat exchangers, and so forth), from which the user has to select the ones relevant to the situation (scenario) for which a model has to be constructed. CyclePad works in two phases, build mode and analyse mode. In the first phase (build), students use a graphical editor to place components and connect them. Figure 4.2 depicts such a structure. In the second phase, the analysis mode, the working fluid and modelling of the components are specified, and numerical values for the properties of components are then entered by the user. As soon as this information is provided, all performance
4.2. RELATED EDUCATIONAL SOFTWARE

Figure 4.2
CyclePad User Interface.

characteristics of the system are displayed. At any point questions can be asked by clicking on an item to obtain the set of questions (or commands) that make sense for that item. Remodelling the current model may be necessary when the behavioural simulation does not produce the desired results.

4.2.2 Predefined Simulations

Using Predefined Simulations is probably the most common approach to simulation-based learning (e.g. ITSIE [103], and SimQuest [72]). Systems that employ predefined simulations have in common that the simulation is not created by the learner. Instead, the model is built by the developers of the interactive simulation and the learner can merely interact with the prefabricated simulation. This interaction can take many specific forms (see e.g. [31]), such as answering predictive questions, controlling the simulation (as in a flight simulator), setting up experiments and discovering what happens, finding explanations for observed behaviours, and many more. The idea is that by interacting with the simulation, possibly taking into account a set of assignments, a learner will eventually understand the behavioural insights captured by the simulation and thus understand (at least partly) the real behaviour of the system. In the following we discuss the systems ITSIE, SimQuest and Co-Lab which encompass rich manners of providing support.

ITSIE

ITSIE (Intelligent Training Systems in Industrial Environments) is a learning environment based on multiple qualitative and quantitative models. In this system, simulations can be used in two modes: free exploration and apprenticeship. The authors claim that the advantage of this approach "is that it enables the teaching of multiple models of a physical
system, teaching not only the models themselves, but also their strengths and limitations and how each can serve a different purpose during problem-solving" [103].

The simulations in ITSIE are required to be ones in which users such as engineering students can interactively control and observe the resulting behaviour. The interaction between the learner and the simulation is made up of a combination of case reviews, the manipulation and observation of the behaviour of the system using a variety of models, and browsing through a collection of provided instructional material. The instructional material can either be browsed by the learner directly or an instructor can guide the student in their use.

**SimQuest**

SimQuest is an authoring system for designing and creating simulation-based learning environments [33], in which simulation models form the core of the learning environment. This authoring environment is designed to support experts in building applications and thus it is not a modelling environment to "learn by building models". Therefore, we do not consider the authoring task in this analysis.

Learners interact with the model created by an expert, usually a teacher, through an interface also created by that expert. The interaction usually consists of changing values, for example by moving sliders, clicking on radio button, setting dials, etc. The interface directly reflects the behaviour of the model in response to these changes by means of graphs, dials, thermometers, etc. Figure 4.3 shows a screenshot of an example system.

![Figure 4.3](image-url)

*Figure 4.3*

SimQuest: Example of a Learner interface.
4.2. RELATED EDUCATIONAL SOFTWARE

Co-Lab

Co-Lab provides a Web-based environment for collaborative inquiry learning [98]. The system is designed for learning natural sciences at the upper secondary level and the first years of university. At the time of writing of this thesis, four domains were available: water management, the greenhouse effect, mechanics and electricity. Co-Lab is designed on the basis of a "building" metaphor. Within each building there are floors and rooms. Each floor implies a different level of difficulty. On each floor students have access to four rooms (Hall, Theory, Meeting, and Lab) which correspond to the processes of exploration, hypothesis generation, experimentation, and evaluation, respectively. Each of the rooms has specific tools which students use to conduct different aspects of their inquiry.

Like in SimQuest, there are two groups of users of the Co-Lab environment. The first group are learners, who interact with the experimental space by working on tasks assigned to them by a teacher. The learners collaborate on experiments, create models and explain the events in the experimental space. The second group are authors, typically teachers and instructional designers, who configure the environment to be used by learners. However, the tools addressing the latter are designed for expert users and not for learners who learn by building models. Hence, this modelling environment is not further discussed here.

4.2.3 Intelligent Tutoring systems

According to Halff [55] a tutoring system must exhibit the following three characteristics. It must exercise some control over the curriculum, that is, the selection and sequencing of material to be presented. The system must be able to respond to questions of the students about the subject matter. The system must be able to determine when students need help in the course of practicing a skill and what sort of help is needed. Furthermore, in order to be intelligent the system should be knowledgeable about the subject matter (or domain) to be able to make inferences or solve problems related to the domain, and additionally it should be able to diagnose the knowledge of the students and cognitive characteristics. Furthermore, it should also include knowledge about teaching strategies.

Below we discuss three ITS systems: Andes [115, 53], EUROHELP [20] and STAR-Light [37]. Each of these systems employs different approaches to interacting with users.

Andes

Andes is an intelligent tutor that allows students to solve Newtonian physics-related problems in an environment that provides "visualisation, immediate feedback, and procedural and conceptual help" [115, 53]. A Graphical User Interface (GUI) supports the student in making drawings that represent the problem to be solved. Within this GUI, the student may also define variables, enter the relevant equations and obtain the respective numerical solutions. The interface provides a set of icons that can be used to construct free-body and motion diagrams. Vector quantities and variables are also represented iconically. The environment includes provision of help for the solution to the problem at hand if requested by the user.
The graphical user interface is divided into three areas, Figure 4.4: a pane containing the problem's textual statement and a drawing illustrating it (students are allowed to complement this drawing); another pane (top right pane) showing the defined variables; and finally a pane for the students to enter their equations (bottom right pane). To interact with the help system, the user has to click on the respective help button. A dialog box featuring hyperlinks for accessing further explanations is then displayed.

![Figure 4.4](image)
The Andes System.

**EUROHELP**

EUROHELP [20] is a shell for building intelligent help for software applications. Using the available tools and following a defined methodology, designers of Help Systems can feed the shell with information about a specific application for which a help system needs to be created, for example a text editor, a database system or a graphics program.

One of the prototypes implemented in the context of the EUROHELP project was help for the UNIX e-mail program. There, the interactions with the user were of a known and straightforward nature. It used natural language dialogue facilities and provided support in automatically generated natural language. User interaction was effected via command line.

**STAR-light**

The STAR-light system is a prototype implementation of the STAR² framework [37]. The STAR-Light prototype applies model-based reasoning to the design of educational

²System for Teaching About Reasoning.
software. The interface of the prototype consists of displaying pictures of the behavioural states of a system, presenting multiple-choice questions and providing explanations whenever a wrong answer was detected or the so-called 'give me hint' button was pressed. The interaction with the system is individualised since the program builds a model of the learner and compares it to a normative model for the subject matter domain. It implements a question generator and an explanation generator, producing questions and explanations about the behaviour of some (physical) system, and is used to interact intelligently with a learner. Whenever the behaviour deviates from the norm, the software develops an appropriate diagnosis and applies an appropriate pedagogical intervention, which takes the form of presenting new questions to the learners.

4.2.4 Software Agents

Agents are ideally suited for hiding the complexity of difficult tasks and performing inessential, laborious tasks on behalf of the user [78]. They may additionally be used to train or teach users, help different users collaborate, and monitor events and procedures [78]. A variety of software agents has been developed to assist users. The set of tasks or applications an agent can assist the user with is virtually unlimited: automating desktop tasks [58], mail management, meeting scheduling, selection of books, movies, music, and other forms of entertainment [78], electronic commerce [41], information filtering and information retrieval [23, 107, 76, 86], virtual environments [64], computer games [112], and interactive learning environments [12].

The original conception of an intelligent agent was something like a personal butler or assistant [82] and many research programs implemented agents with those capacities for the purpose of assisting in learning. However, researchers claim that in order for an agent to be an effective mentor for learning, it should include other main requirements, such as, regulated intelligence and the existence of a persona [9, 60]. Below we focus on the personified aspect of agent technology for use in ILE\(^3\). In what follows, different approaches in the employment of software agents are briefly presented using representative systems as examples. The first two example systems use non-animated characters for interacting with users. The first one of these, Betty’s brain [12, 74], presents one main character providing support. The second one, SCI-WISE [102], uses a set of different characters, each one supporting specific tasks according to its expertise. Finally, Adele [101], an additional example of an agent system, is presented which has the characteristic of interacting with users by means of an animated agent.

**Betty’s Brain**

Betty’s Brain [12, 74] is a learning-by teaching environment. In this kind of environment, authors use AI-techniques to create teachable agents, and it is the responsibility of the students to teach the agent. Thus the teachable agent has no automated learning algorithms built into its learning process, it needs explicit instructions to perform well. In Betty’s

\(^3\)Notice that aspects about contents or intelligence are addressed in the two previous sections of this chapter.
brain the task of students consists of teaching the computer agent Betty by building a concept map that models relations between domain concepts. The relations can be causal, hierarchical or property links between the entities that represent the domain. The primary component of the agent is its decision-maker. It incorporates the qualitative reasoning mechanisms for generating answers to queries from the concept map structure, as well as schemes that implement strategies governing the dialog process with the user. Students use a graphical drag and drop interface to create and modify their concept maps in the top pane of the application’s window. Students can query Betty at any time using an Ask button. She will then provide an explanation of how her answers were derived by depicting the derivation process using one or more of multiple possible modalities: text, animation or speech. Figure 4.5 illustrates the interface of Betty’s Brain.

![Betty’s Brain Interface](image)

**Figure 4.5**
Betty’s Brain Interface.

As part of the teaching process, students can query and quiz Betty to assess her understanding based on what she has been taught. Students ask her two types of questions: (i) "What happens to <concept A> when <concept B> increases/decreases?" and (ii) "Tell me about <concept A>". Betty answers questions by employing a qualitative reasoning mechanism, and explains her answers verbally and by using animations. The qualitative modelling framework uses qualitative mathematics, with tables for composing discrete values to provide to the qualitative simulator.
ThinkerTools SCI-WISE

The SCI-WISE system [102] incorporates a whole community of agents that give strategic advice and guide middle school students in the process of acquiring inquiry skills and "reflective, meta-scientific ability to build general theories of the inquiry process and learning in general" [119]. Each agent has particular areas of expertise, with general-purpose agents such as "Ingrid the Inventor", or task-specific agents such as "Quincy the Questioner" or "Helena the Hypothesiser". The system supports inquiry learning by asking a student to explicitly state the purpose of each action and provide a justification for it. Each agent's expertise is packaged as space of HTML documents, some of which can be "activated" by a production system contained within the agents. The advisors have general domain knowledge. For example, the domain of the Questioner advisor is research questions, such as criteria, strategies, examples, concepts and referrals to other advisors. An example of a criterion is What makes a good research question?, that of a strategy is How do we come up with a good research question? Furthermore, the advice is contextual. This means that to give advice, the advisor looks at the task that the student is working on and the reasoning engine, which is rule-driven, establishes beliefs about which advice or message is appropriate.

The student develops his/her own theory of scientific inquiry by modifying the agents so that they reflect the student's beliefs. To do so, the system includes a suite of graphical tools to assist in such modifications. The interface is structured around a tabbed pane interface, to which the agents can add sections and pages. The pages mostly contain text areas for the user to type into, but they may contain tables and lists of text as well.

Students work in a GUI, called Research Notebook which has a page for each step in the Inquiry Cycle. Each page has several workspaces, each of which typically corresponds to one step in the plan that is recommended by the advisor who governs that page. In the Research Notebook students develop their research question, generate hypotheses, design an investigation, record and analyse their data, create models, and evaluate the utility and limitations of their models as well as their research processes.

Adele

Adele [101] supports medical students in achieving clinical skills. The agent, Adele, is a representation of a doctor that assists the user of a Web-based medical simulation environment. In a typical session of Adele students are presented with a computer simulation of a clinical problem. They are able to examine the simulated patient, ask relevant questions, order and interpret diagnostic tests, and make diagnoses or create treatment plans. Figure 4.6 shows an example of a student session. The agent is able to monitor the progression of the user in solving problems, gives feedback and evaluates the performance of its users. Adele can suggest on-line medical resources that are relevant at the current stage of the case work-up. For example, when a student selects a diagnostic procedure to perform on the simulated patient, Adele may point the student to video clips showing how the procedure is performed.
4.3 Lessons learned regarding interface and interaction design

The analysis of the system categories described above revealed a number of Graphical User Interface (GUI) features that make their use more effective.

In an educational context a GUI may provide additional assistance by taking over certain routine parts of the problem-solving task or by making a task seem more manageable merely through an improved visual presentation. In fact, when properly implemented, this may lead students to concentrate more on the high-level structure of the task without being overwhelmed by the often uninteresting details [22]. This will lead to the ultimate goal of a more efficient learning curve.

In what follows, we will pinpoint those GUI features on a per category basis, compare their settings with the HOMER environment and finally try to define how they can be included in the design of a support module for a new model-building environment.

4.3.1 Design Environments

Most of the systems studied in the above presented design environments are part of university curricula and as such chances are high that the typical users are already familiar with the employed vocabulary, which facilitates the implementation of the model-building task. Therefore, these software packages are incremental aids intended to complement the teacher's classroom work.

In HOMER this is not the case. Except maybe for courses in Artificial Intelligence, Qualitative Reasoning is often not part of any standard curriculum. In the particular case

Figure 4.6
Adele's User Interface.
4.3. LESSONS LEARNED

of the experiment performed during the course of this project (see Chapter 3), the employed iconic language and its meaning as well as the underlying domain were new to the tested subjects. Using HOMER thus requires users to either previously learn the HOMER vocabulary or learn it while using the tool. As a result, in order to address the lack of prior knowledge, an improved version of our QR workbench should incorporate the necessary online help to explain concepts concerning QR as well as provide assistance related to the use of the tool itself, e.g. the meanings of icons and the necessary sequential steps for building models. Models in CyclePad are assembled from predefined parts, which are represented by icons that are directly associated to their real world counterparts. This is an advantage, however it is restricted to the specific domain of thermodynamics. Furthermore, the students learn about the real world components before they use the software. We have chosen to develop a domain-independent environment thus the icons employed are general ones and in principle students cannot immediately associate the underlying concepts with their visual representations.

Two main features of the user interface design of STELLA that we believe will be an added value in the design of our new tool are the provision of multiple representations of the simulation results and the integration into one system of the model construction and simulation parts. HOMER does not include the latter and as such, if the user wants to run a simulation of a model, the user needs to export the model, open it in the simulator, which is a separate program, and start the simulation. The simulation outcomes play an important role in the learning process. They provide feedback to the modelling activities. For the modelling task to be more meaningful, thus, the new implementation of our model-building environment should also integrate the simulation aspect thereby linking the simulation to the model.

CyclePad's interface clearly distinguishes two phases of the user task, namely the build and the analyse phase. Users are always made aware of the currently active phase. It is quite important to have these clearly delimited phases. During the evaluation of HOMER, subjects encountered problems in terms of the logic of the different stages of the model building process. For instance, building a scenario was too similar to the task of building a model fragment. To alleviate this problem, the interface should be designed so as to support a clear distinction between the different steps composing the model building process.

4.3.2 Predefined Simulations

When using predefined simulations the task of the learner is facilitated by the fact that there is no need to learn a new language either to interact with the simulation or to interpret the simulation results (shown using traditional means) as is the case in our approach. From this point of view, when comparing these systems to HOMER it is clear why the latter is more difficult to learn. This is a problem that needs to be addressed by providing adequate support.

All known applications created with SimQuest use drawings to illustrate some part of the assignment. Research shows that providing this type of external representation facilitates cognitive tasks such as problem solving, reasoning, and decision making. They
provide information that can be perceived directly and can be used to anchor cognitive behaviour [121].

During the experiment with HOMER, subjects spontaneously made their own drawings on a piece of paper, sketching the problem to be modelled. In fact, many scientists use diagrams or other forms of external visualisation to aid their thinking. Researchers have proposed that diagrams have a profound effect on scientific reasoning [51]. As in our approach the user is expected to construct the whole model autonomously, it would be helpful if the interface would be equipped with a drawing tool. Having a drawing tool instead of letting the modeler use pencil and paper is important because of the enhanced ease in manipulating drawings on screen provided by the former.

Another important feature of applications created with SimQuest is the so-called 'monitoring tool'. With this tool learners can store results from different simulations, sort them and display them side by side to compare the results from different experiments. Similar approaches in providing overviews of actions of a learner have been used in the past [92, 32, 75]. Although we do not intend to implement a similar feature in our new design it is an interesting one to keep in mind for future research.

From Co-Lab an interesting feature is the implementation of collaborative work. Students work on shared information and also have access to a chat tool. Indeed Computer-Supported Collaborative Learning (CSCL) is a well established paradigm of educational technology and several research topics have emerged in the context of CSCL [38, 69, 87]. Having such a feature in a model building environment would enable modelers at different physical locations to work together on model construction and simulation.

### 4.3.3 Intelligent Tutoring Systems

Of the three ITS example systems presented above, the only one we will analyse here regarding GUI design is Andes as it is the only one the employs a graphical user interface. Icons in Andes are used for drawing traditional diagrams that are also typically used in textbooks, for instance motion diagrams. These are familiar to most students and therefore no new iconic language needs to be learned. Similar to what was suggested in the discussion on the Design Environments we will need to compensate this lack of knowledge in our approach by providing support regarding the modelling language used.

In all of the problems proposed by Andes, the student is encouraged to make drawings (drawing a body, drawing forces, drawing a coordinate system). Also, the assignment that is given to the learners is part of the environment and is explicitly shown, making the user constantly aware of the goal to be reached. Furthermore, for all possible actions, like making drawings, defining variables to be used and entering relevant equations, users receive immediate feedback in the form of their entry turning green when correct or red when incorrect. This feature prevents students from wasting time by following incorrect paths in their solutions. When students request help, a dialog box with advice is displayed. These dialog boxes contain further options and if a student wishes more specific advice he can press one of the hyperlinks "explain further", "how" or "why". In HOMER, as in Andes, users are prevented from constructing erroneous models and we consider it to be a positive feature which should be kept. The support in making drawings is a feature
4.4. FEEDBACK ON ERRORS

we want to implement as well. Furthermore, the way that Andes presents advice to the students is simple and effective and would apply well to our improved system. Finally, hyperlinks represent a useful feature that has been used frequently in online help systems and we intend to implement it in the improved version of our model building environment.

Finally, all three ITS systems described above have several attributes related to individualised user feedback which are important achievements in the area of student modelling and therefore have a close connection to the support needed in HOMER. The issues of feedback on the task of the users will be discussed in Section 4.4.

4.3.4 Agents

The approaches taken in the systems discussed in this category give us ideas on how to design our support module, especially concerning the form in which help is presented to the user. From SCI-WISE we learn that having different agents with specialised types of skills is an interesting manner of providing highly organised support which helps the learner to stay focused. Furthermore the personification in Betty’s Brain and Adele re-inforces the claim that the use of agents increases motivation and as a result their users learn more effectively. Therefore, personification and modularity should be considered in the design of our support module.

4.4 Feedback on errors

Within the scope of this thesis we are particularly interested in ways of providing help to users trying to accomplish model-building tasks. For that reason, our analysis emphasises educational settings. In the previous sections we have analysed a number of selected systems and looked at the interface features intended to effectively facilitate the tasks of the learners. In the following paragraphs different approaches for providing user feedback are discussed. We then look at how those approaches relate to the design of our support module, Section 4.5. Later on Section 4.6 discusses other complementary approaches for supporting the task of the users.

4.4.1 No Feedback

Some of the above-mentioned systems do not include user feedback. STELLA is one of those systems. It does not possess knowledge about the scientific domain it is being used to model. As such, the tool can provide little or no assistance to the user in making semantically-based modelling decisions – for example, decisions about which modelling equations are appropriate for solving a given modelling task. Building models implicitly includes filling out many technical details in terms of the underlying mathematical equations. Certainly, providing the model building blocks and automatically extracting the associated equations already presents a great aid to the modeler.
4.4.2 Predefined Feedback

Usually in interactive learning systems where the user has to complete an assignment, the feedback provided on the occurrence of errors is hard-wired. For instance, for the systems falling under the predefined simulations category (see section 4.2.2), it is mostly the case that the learner has to answer a series of (multiple-choice) questions after the completion of a task. When a question is answered erroneously, an immediate, usually predefined form of feedback is given.

4.4.3 Tailored Feedback

A more sophisticated form of user support comes in the form of feedback tailored to the specific needs of the user and/or the situation. Intelligent Help Systems, such as EUROHELP, are meant to understand what tasks a user is trying to perform and then give feedback accordingly. In EUROHELP, user actions are constantly monitored with the intention of inferring the goal the user is trying to reach. The help system contains a library of correct as well as incorrect and sub-optimal plans. It tries to match the sequence of user actions against these plans in order to find the one that fits best. EUROHELP provides natural language generated help and contextual information. Furthermore, the user can ask questions at any point and answers will be given in terms of the specific current context.

The STAR framework, on the other hand, constructs its diagnoses using models that seek to "identify those sets of primitive reasoning steps whose correct behaviour is inconsistent with the observations" [37]. A diagnosis is performed at the level of problem-solving behaviour and thus it does not attempt to build a learner's mental fault model. In many other intelligent learning applications the diagnosis of the problem is based on elaborate, detailed, and expensive "bug catalogues", which try to include all known ways in which people make mistakes in a given domain, and then compare the items in the catalogue to the learner's mistake. By using model-based diagnosis, STAR does not require explicit fault models or bug catalogues for its operation, but instead reasons on the basis of a representation of the correct behaviour of the system to be diagnosed. The added value of this approach lies in the generality of the method. As long as the subject matter complies with the modelling principles, the same diagnostic component may be applied to different subject matter models. Thus, the system focuses on learning from errors: "instead of directly mapping errors onto misconceptions to be remediated; zooming in on the specific bug that causes the error guides the learner in discovering this error and maybe self-repairing it" [37].

STAR-Light asks students to predict what will happen in a selected example situation from physics. When learners make a false prediction, the software probes their knowledge with several multiple choice questions designed to find out where the exact problem lies. During a series of experiments, the prototype was successful in finding errors and correcting them. In fact, more often than not, learners realised their own mistakes during the probing process.

Andes allows students to follow different correct solution paths to a problem, and
to skip steps on their way to the final solution. This makes it more difficult to understand what a student is trying to do at any given point, and to track the corresponding knowledge. To cope with this, Andes uses Bayesian Networks to perform knowledge assessment. The Bayesian Network based student model is also used in Andes for plan recognition, to figure out a goal of the student during problem solving and suggest steps for the student to achieve that goal. Thus, feedback is individualised with respect to the plan of the student. The model is also used to adjust the way in which Andes presents help when it decides that the student has difficulties in the use a specific rule of physics [53]. By consulting the student model, Andes presents in detail those rules which the student is not familiar with.

A somewhat different tailored feedback is given in CyclePad. In CyclePad, students can use the *e-mail coach* to receive feedback on their design. It provides additional analysis help and uses analogical reasoning to do the coaching. The email coach makes suggestions on how to improve the design of the student, based on analogies with a web-based library of expert-authored cases. It provides step-by-step instructions illustrating how this suggestion might be applied to the particular design of the student. It does not, however, evaluate whether or not this suggestion is a real improvement.

### 4.4.4 Feedback based on user input

In contrast to the feedback provided as a reaction to the actual problem-solving, another approach for learners to acquire new knowledge is by giving feedback in the form of showing the consequences of the input of the user. Thereby the feedback is targeted at the meta-cognitive skills of the learners.

For instance, in *Betty's Brain*, students see the consequences of their teaching, and thereby implicitly deduce the state of their knowledge, when the agent Betty has to answer questions about the subject being taught. In *SCI-WISE*, students develop their own theory of scientific inquiry by modifying the agents so that they reflect beliefs of the students. In order to test their theory, students then engages in a physics project, during which agents provide advice. Students evaluate the agents behaviour, which may result in the modification of the theory.

Simulations that can be initialised and run also are a form of this type of feedback. The output produced by the simulation is based on the input of the user and as a consequence the user can analyse and verify the correctness of the input he provided.

### 4.4.5 Feedback based on Example, Counter-Example and Analogy

Examples are an important means of providing help to a user because they focus his/her attention on how best practices, designed by experts, may be used for solving a problem/task efficiently. In addition, studying examples is a natural and common way of learning, and students are used to studying examples extensively when acquiring new skills [88, 114]. Moreover, the use of examples and counter-examples helps students better understand not only what to do, but also what not to do and to what degree. Examples can be used to provide user support in two ways. One way is directed towards showing how a chosen
example has key characteristics in common with the problem at hand. A second sort of support focuses on showing how newly introduced concepts can be instantiated in a given example.

The use of analogies and examples is an often used way of creating an understanding of a novel domain. Organising the understanding of a domain by imposing on it an already familiar structure can be a highly efficient approach to understanding it [24]. The ability of humans to understand novel situations by analogy to familiar ones and to solve new problems by remembering solutions to previous, analogous ones, has motivated the study of analogy as a learning theory. According to [50] analogical reasoning is central to problem solving, learning, and creativity.

Analogical reasoning identifies certain features in new and old problems and tries to use the old solution to infer a new one [50]. In fact, when teaching a new concept, it is normal practice to use analogies to familiar or previously learned material from diverse domains. For example, "an electric circuit can be understood by comparing the causal relation between voltage and current to the causal relation between pressure difference and flow rate of water". In an analogy, similar relations between two domains are matched or placed into correspondence. Thus the domains are aligned based on a shared relational structure [52]. Individual elements in the two domains might be semantically quite different, but the structural relations between those elements is the same in both domains [24].

4.5 Lessons learned regarding feedback on errors

In order to illustrate how the analysis of user support/feedback presented in the previous section can be used in the design of our new model building environment, we now compare how the feedback provided in the above-presented systems and categories relates to HOMER. The focus of our discussion is on how the particular features can or cannot be used in the design of a support system for our type of application.

By remaining domain-independent we cannot easily provide the same type of support as given in other systems. Deep knowledge about the whole spectrum of known scientific domains would be required as well as an extended library of alternative ways of modelling the same problem. Thus for the reasons just explained, semantical correctness cannot be guaranteed by our system.

Intelligent Tutoring systems (and also the Adele agent system), demand a high degree of knowledge engineering effort, since many different kinds of knowledge are needed. As such, knowledge about the problem-domain is of great importance, but also knowledge of typical problems users face when interacting with the system. In the case of systems featuring a natural language dialogue interface, knowledge about dialogue and discourse is also relevant [42]. Depending of the goal of the system, knowledge about teaching strategies may also be needed. Given these extensive requirements, we do not intend to implement a full-fledged ITS. We can however provide intelligent support to some extent. We can use model-building characteristics and constraints as well as the way in which experts usually build their models (best practices) in order to coach the user.
From the agent paradigm we can benefit in two ways. Firstly by breaking down the type of support that may be given to a user, similar to what has been done in the work of [102], thereby making the step of looking for help a more focused one. For instance, we may have an agent responsible for giving answers regarding the model building ontology, one for giving explanations about the graphical language and another one responsible for explain the procedures to complete a specific task, in a manner similar to the agent structure implemented in SCI-WISE. Also, just as was done in Betty's Brain, we can allow users to teach an agent how to model a system and then, when new users try to model a similar system, the agent can demonstrate how that problem was previously solved. Animated agents are a great promise, however we believe that in a first instance the quality of the help contents should be tested before engaging in a more computationally expensive approach.

4.6 Support Designing for the Learning Process

In this Section we raise some complementary issues which we consider to be interesting when approaching the design of support. They relate to how the knowledge to be communicated/taught can be better organised and subsequently be more smoothly presented to the user.

4.6.1 Organising the Domain Knowledge

Teaching a new subject matter involves a set of important characteristics. Firstly, attention must be paid to basic ideas and principles. These basic ideas and principles will establish a solid base for later learning and understanding as one moves into more complex operations. Secondly, understanding the interconnections among these concepts is relevant. As a result, instead of learning isolated topics, students will acquire a unified body of knowledge. Thirdly, one must acquire the capability of looking at a problem from multiple perspectives to find alternate ways of stating a problem and finding a solution. Certainly there is a great variety of strategies for solving a specific problem. The theory of model progression [118] relates to this argument. This theory specifies a progression of increasingly complex models that students should master while learning about the behaviour of physical devices; learners are scaffolded in a stepwise orientation on the variables that are part of the simulation model.

This applies to building models as well, since there is a definite difference in problem complexity between constructing a simulation model with a few objects, quantities, behaviour states and so on, when compared to building a full-fledged simulation model with many objects, quantities and behaviour states. In general, the more model ingredients are involved, the more difficult it is to understand the simulation model as a whole.

In fact, model progression is a key feature for predefined simulations, otherwise the simulation models would become unwieldy. SimQuest, for example, enables the implementation of model progression in order to support students. In HOMER it is less of an issue due to the fact that users will build their own simulation models. However, if HOMER is going to be used in a classroom, one, for instance, the teacher, should
definitely make sure that assignments given to the modeler are reasonably manageable, growing from simple to more complex models. However, the generation of assignments is not part of HOMER but something that the person who gives the assignments should take into account.

4.6.2 Conceptual and procedural knowledge

A typical way of dividing knowledge from a learning perspective is by organising it into classes of declarative and procedural information [4]. In fact, what people most typically do is apply both declarative and procedural knowledge to accomplish a particular task. Therefore, it is wise to design the contents of a help system as to include both types of knowledge. In what follows we will briefly explore this knowledge distinction and mention its utility within a help system as well as give some examples of its use.

**Conceptual knowledge** In order to cope with the world that surrounds them, humans have developed ideas, insights and theories concerning things, what they are and how they are interrelated. Persons entering a new arena need to learn this in order to be able to share the knowledge the community has developed. The concepts must be explained to them in order to enable them to understand the domain. Therefore, providing help on conceptual (a.k.a. declarative) knowledge refers to realising knowledge transfer concerning basic principles, ideas, insights, theory and so forth. It is knowledge that is factual in nature and can be made explicit. It provides the "what" knowledge which is needed to complete a task and thus describes (explains) specific concepts or features, such as what is a *quantity*.

To render this idea more explicit, imagine a problem involving the modelling of a parachute slowing down the descent of a person. When confronted with the phrase "...the acting forces on the parachutist are the gravity...", users may not realise that the gravity is pulling the person downwards and thus is exerting a force on the person. They may not have understood the meaning of the word "gravity". Using the above categorisation, explanations regarding the concept of "gravity" would be classified as conceptual knowledge. In summary, conceptual help includes the knowledge (explanation) about the specific domain of the modelled system, e.g., physics, statistics, biology, medicine, architecture.

**Procedural knowledge** Procedural knowledge on the other hand contains detailed knowledge about how to accomplish some task focusing on information regarding the "how" rather than the "what". It refers to the execution of steps to perform a task. In essence, procedural knowledge is about using conceptual knowledge to achieve tasks. For instance, somebody may know the concepts of screen size and pixels, and colours but, knowing the steps needed in order to change these properties under a given operating system is another thing. Different systems (e.g. Windows and Macintosh) may require different steps to perform this task. Help can always be provided on how to do this exactly and more efficiently. An ordered list describing a sequence of steps like for instance mouse clicks on a GUI in order to change these
properties would then be classified as procedural information. Procedural help does thus not directly cause users to master a specific domain.

The *optimal* help system would be one supporting both kinds of knowledge: conceptual and procedural knowledge.

### 4.7 Our approach to the new support system

As we have learned from the analysis on the various systems presented in previous sections, opting for a domain independent approach leads to a number of consequences concerning how to effectively provide support to users. These consequences can be summarised as follows:

- Besides providing support to the users in acquiring conceptual knowledge about the QR ontology, support concerning the graphical language should also be given. By being domain-independent, the icons employed in HOMER and also in our new workbench are general ones and in principle users do not immediately associate the underlying concepts with their visual representations.

- We cannot implement user modelling in the sense of creating a cognitive model of the knowledge of the user. There are two reasons for this. Firstly, the amount of user freedom offered in a general model-building environment is so large that a full learner model is beyond the scope of a practical application. The number of alternative ways in which a situation may be modelled is simply too large. Secondly, often, user modelling is seen as contradictory to the concept of constructivism, for which *measuring* the knowledge of the learners conflicts with the idea that each learner builds his or her own representation of the external world (for a discussion on this point see, for example, [65]).

Table 4.1 summarises the lessons learned from our previous analysis. As we cannot provide support based on the knowledge of the user, our goal then is to develop a support module that supports users in the process of building models based on their individual interaction with the learning environment. Thus, the input of the user (the model created or partially created) is used as a basis for generating tailored advice.

HOMER already includes several mechanisms which facilitate the task of the users. Firstly, it consists of a *set of builders*, each one serving the purpose of constructing knowledge specific to a clearly delimited subtask of the overall model-building process. It is a well documented fact of human cognition that the decomposition of large tasks into a set of nearly independent subtasks is a way of reducing cognitive load for the user, thereby reserving adequate capacity for learning [7]. Secondly, HOMER uses diagrammatic representations [70] of knowledge, which makes the structure of a model more visible and more easily manageable. Thirdly, the user interface is built in a manner that prohibits the construction of syntactically incorrect models.

However, building qualitative models is a complex task [100] and our experiment with HOMER has shown that the support provided by HOMER is still insufficient. For that
Chapter 4. Enabling Support in Building Models

### Type of Feedback

<table>
<thead>
<tr>
<th>Type of Feedback</th>
<th>Content</th>
<th>Form of presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predefined</td>
<td>Background Knowledge: QR Ontology</td>
<td>Agent-Based</td>
</tr>
<tr>
<td></td>
<td>Graphical Language</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Declarative Knowledge)</td>
<td></td>
</tr>
<tr>
<td>Tailored</td>
<td>Guide the users in the good practices in building models, for instance, by analysing the model constructed by the user and then providing guidance concerning what the user can still do and also pointing out incompleteness in the model.</td>
<td>Agent-based</td>
</tr>
<tr>
<td>Feedback on the input of the user</td>
<td>Providing means for running the simulation of the model and inspect the results of the simulation</td>
<td>By means of the user interface</td>
</tr>
</tbody>
</table>

Table 4.1
Requirements for our support module.

reason, an improved workbench providing adequate support has to be. This workbench is further described in Chapter 5. In this section our focus is on how feedback can be given to users during the model building phase.

As we have learned from the analysis of the systems presented earlier, rather than just supporting users in "performing tasks", the support system of the new version of our model-building tool should provide the more advanced concept of "learning while doing". Furthermore, support should be presented in a motivating manner to the target audience to invoke its frequent use.

The aim is twofold, to help users in acquiring a deeper understanding of the concept being modelled and at the same time improve their modelling skills. Notice that this distinction is similar to the notion of conceptual and procedural knowledge mentioned earlier.

Since we opted for a domain-independent approach to our modelling tool, some constraints on how we can support the user have already been set. As the conceptual knowledge does not change, it may be designed as Predefined feedback, which is rather static. Conversely, providing support on procedural knowledge in domain-independent systems can be individualised since it refers to the modelling rather than the underlying domain. It is thus possible to compare the user's model-building actions to common practice in modelling as defined by expert users and provide tailored feedback in order to guide the user towards acquiring improved modelling skills. This kind of support will be of a more dynamic nature.

Following the above categorisation, which makes a distinction between static and dynamic support, we will next analyse the design details of our support system.

### 4.7.1 Static Help

Some of the problems encountered by the participants during the experiment with HOMER originated from not understanding or not remembering definitions concerning the model building ontology. This was by far the main source of difficulties. Problems
such as differentiating between attributes and quantities or grasping the meaning of a dependency were quite common. The support system should thus particularly stress this issue.

As explained above, support referring to conceptual and procedural knowledge will be included in the static help category. We have opted to implement the conceptual support in two complementary forms. Firstly, by providing definitions for the terms composing the model-building ontology and secondly by giving examples on how to use those terms. Procedural support will explain in detail how to complete a given task using the underlying user interface. The static help system should thus be able to answer questions such as what is an influence? and explain how to create an influence using the available tools.

4.7.2 Dynamic Help

Besides providing guidance regarding the ontology and the use of the model-building tool, the help system should also provide support with respect to the specific content of the model being created. This type of help needs to have assessment capabilities concerning the prior and actual production of the user in order to be able to evaluate the progression of the user. Since this progression is a dynamic process, the contents of the provided help will be constantly changing.

The dynamic support system continually analyses the current solution of the user to the assigned problem and compares the steps taken to reach this point with a selection of correct modelling features. Any inconsistencies will be detected and reported to the user so as to instigate the user to reflect on the actions taken and maybe consider an alternative trajectory. By doing so we try to keep learners on track so as to avoid them arriving at incomplete models.

The dynamic help system is designed to provide guidance at two distinct levels, one regarding local, the other global knowledge. The former will be concerned with the details of a specific modelling subtask and will usually be restricted to a certain builder. The latter, on the other hand, will give a global perspective on the modelling activities of the user, reuniting the actual status of the full model, so as to synthesise the sometimes overwhelming complexity of large models. This distinction between local and global knowledge is an important one, since the construction of models from scratch will usually be a constant interplay between figuring out the fundamental details of the underlying model ingredients and defining the overall interrelationships between those ingredients.

Local Analysis  At this level, help is generated on the actual model-building activity of the user. The help facility analyses the input of the user within the active builder and guides the user by providing a set of possible subsequent actions. Also context-sensitive help is given and focuses on the specific request for guidance from the user. For instance, if the user selects a quantity in the model fragment builder and then select Local Analysis help, only guidance regarding that primitive will be given.

Global Analysis  Global feedback on the other hand is based on what the user has previously constructed in all existing builders. We define a progress level in order to
generate contextual advices associated with each model building step. Thus, the information gathered enables the help engine to build a structured list of possible user actions applying to the model building step. Using the "construct a scenario" task as an example Table 4.2 shows an ordered list of such progress levels. The third column gives the general meaning of an advice; the fourth column gives an explanation about the advice\(^4\) and also suggests possible actions. Each piece of advice will fire if the user is at or further than the corresponding progress level, but has not reached the next level yet.

<table>
<thead>
<tr>
<th>Progress level</th>
<th>Advice Number</th>
<th>Advice</th>
<th>Associated piece of explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>1</td>
<td>Scenario X is empty</td>
<td>The scenario &quot;X&quot; is empty. Nothing has been specified yet. You can start filling the scenario by adding an entity.</td>
</tr>
<tr>
<td>An Entity</td>
<td>2</td>
<td>Missing quantity</td>
<td>The entity &quot;Entity&quot; has not been given a quantity. This may result in the simulation not being able to derive certain behaviour. Maybe, you want to give a quantity to this entity.</td>
</tr>
<tr>
<td>An entity and a quantity</td>
<td>3</td>
<td>No initial values</td>
<td>The quantity &quot;XX&quot; has not been given an initial value. This may result in the simulation not being able to derive any behaviour. Maybe, you want to give a quantity a specific value.</td>
</tr>
<tr>
<td>Two entities and a quantity</td>
<td>4</td>
<td>Missing structural relation</td>
<td>The entities &quot;Entity X&quot; and &quot;Entity Y&quot; are unrelated in this scenario. This suggests that these entities are completely independent from each other. In general, this is not a desirable situation. Probably, you have to add a structural relation that specifies the relationship between these two entities. and Advice 2, 3</td>
</tr>
<tr>
<td>Two entities of the same type with a quantity attached to each entity</td>
<td>5</td>
<td>Missing Inequality</td>
<td>The entities and quantities &quot;X&quot; - &quot;XX&quot; and &quot;Y&quot; - &quot;YY&quot; are of similar type, but they are unrelated in this scenario. In scenarios, inequality dependencies are often specified between similarly looking quantities in order to investigate specific behaviours. So, maybe you want to add an inequality statement between these quantities. and Advice 3</td>
</tr>
</tbody>
</table>

Table 4.2
Progress levels and pieces of advices on the "create a scenario" task.

\(^4\)Earlier advices that are applicable again in a new situation are not fully shown in the table. Instead references are used (For example Advice 4)
4.7. OUR APPROACH TO THE NEW SUPPORT SYSTEM

4.7.3 Structure of the help system

Having described the various contextual forms of help present in the new workbench, we will now describe the manner in which the support information is presented visually and how it can be accessed. As discussed above, the contents of help is divided into static and dynamic information. Since the applicability of these distinct kinds of support is clearly delimited, their availability should also be broken down into discrete stages. Similar to what was done in the work presented in [102] and in order to stimulate the use of help as well as to unambiguously characterise each type of knowledge support, six agents presented as different characters are used. Each agent has a specific appearance representing the type of support it can provide. Figure 4.7 shows these six characters.

Each builder, representing a particular step in the model building chain, possesses its own implementation of the various agents. The whole set of agents is thus present at all times but the support provided will depend on the actual model building context.

4.7.4 Interacting with the Help System

In our approach, the help system will act exclusively on the request of the user. A metaphor that could be used here is that of a student raising his hand and receiving specialised feedback from the chosen coach. By working only on user demand we are in fact adopting a constructive approach to learning [56]. In our approach the user keeps the responsibility for the course of future action and is thus not forced to follow the advice of the agents. This approach is also used in other systems, for example, [85], who use an advisory agent system as a component of intelligent tutoring systems and learning environments.

4.7.5 Agent-based help

Each one of our agents takes the form of an additional tool inside the application. As previously explained, this agent-based help facility is generic in the sense that the agents will display the same functionality for all model building context but the support they provide will be specific to the actual model building step. Moreover, depending on where the user currently is in the model building chain (e.g. building a scenario or a model fragment) the given advice will vary.

Knowledge the agents provide

Static Agents. To answer a question or support the user in solving a problem, static agents use explanatory text, examples and images. The information will be displayed inside a dialog box using HTML pages including hyperlinks and cross-references. Images are also used for displaying GUI parts. Four static agents are included in the new design. They are labelled according to their specific utilities: What is, How to, Curriculum planner and Global help.
Figure 4.7
The support agents.

**What is?** This agent has the task of helping users in understanding model-building concepts in the actual builder. Each builder contains its own version of the *What is?* agent which is equipped to provide conceptual descriptions and examples of any model-building ingredient that can be used in that specific builder.

**How to?** This agent provides procedural help. It suggests the order in which modelling steps should be performed and the actions needed to reach a certain goal. Each builder in the environment has a set of tools for constructing, editing, and removing model ingredients. The *How to?* agent provides the user with procedural knowledge on how to use these tools. It displays only the information which refers to the
4.7. OUR APPROACH TO THE NEW SUPPORT SYSTEM

specific builder being used at the moment.

Curriculum planner This agent is designed to provide help on the curriculum. It is a domain-specific agent, which has educational purposes in mind. Its goal is to provide information related to specific assignments given to students.

Global Help This agent is knowledgeable about general modelling issues. It may answer questions of the type *What is qualitative modelling?* or *How does one build a model?*. In addition, it explains the application of all ontological primitives and discusses basic ideas on how to create a model.

Dynamic Agents As mentioned above, dynamic agents can provide tailored advice and suggestions on both local and global aspects of the model of the user. One of the two agents is concerned with local model building steps and the other is responsible for looking at the global coherence between the production of the user. According to their function, they have been denominated: *What can I do next?* and *Cross builder help*.

What can I do next? This agent gives advice based on the model building ingredients created up to this point in the actual builder. The agent provides intelligent support on the local aspects of the model by hinting on what modelling step can be taken given the current state. The agent also recognises selected model ingredients in the builder and in that case will give advice on the available options related to the selected part. Additionally, the agent informs the user on how to start using a particular builder and reminds the user about essential model ingredients that are still missing in the current model.

Cross builder help The *Cross builder* help agent reasons about the overall model constructed so far and provides support related to the active builder (i.e. the source from where the agent was invoked). The information resulting from the analysis of the complete model is presented to the user in the form of suggestions on how to improve the current model. It may consist, for instance, of pointing out model parts that may be reused or give a list of unused model parts.

Types of advice

The agents have the capability of providing a range of suggestions to the user. However, certain suggestions are more important than others. For instance, making remarks is considered to be less important than pointing out errors. Therefore, it is useful to rank the suggestions in order to determine levels of priority. We use the following ranking in descending order of importance: *warn, advise and inform*:

- **Warn** is about common errors and mistakes. The warning always attempts to present the user with an action suitable to solve a problem. An example of a warning is the following:
Model Fragment is empty!  The model fragment "MF" is empty. Nothing has been specified in either the conditions or the consequences. Maybe you want to start filling in the model fragment by adding an entity as either a condition or a consequence. Alternatively, you may want to add another model fragment as a condition.

- **Advise** refers to suggestions on what to do next, a description of a certain action that may be taken. For instance:

**There are no conditions in "MF".**  The model fragment "MF" contains no specified conditions. That means that knowledge specified as a consequence will always apply. Is this what you want? Or do you want to add specific conditions under which these facts should be applied?

- **Inform** differs from *Advise* in the sense that this type of suggestion does not imply any action, it comes in form of a reminder to the user:

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

### 4.8 Discussion and Conclusions

In this chapter we adhere to the idea that problem solving skills are acquired during problem solving processes, which has been called *learning while doing* [6]. We have also adopted this paradigm, because it represents a constructivist approach to learning. Furthermore, an emerging concept used in new computer based tools for education is the concept of model building. Learners learn by creating, running and adjusting models in a model-building environment. Learning by building models is becoming a popular research area [45]. Similarly, cognitive constructivism emphasises that learning is an active process, and direct experience, making errors, and looking for solutions are vital steps for the assimilation and accommodation of information.

When using the constructivist approach, model building environments offer the learner the possibility of building a contextually meaningful model of the world which the learner can use to improve his current level of expertise. In line with this view, Jonassen argues that students should learn with computers, not from them [66, 67]. Moreover, recent research within the Artificial Intelligence in Education (AIED) community has shown that learners learn more effectively when they are incited to reflect on their learning as opposed to being provided step by step procedures on how to solve a given problem [27, 2]. Furthermore, successful implementations of learning environments have shown that augmenting them with the feature of requesting self-explanation from the users can make the learning process an even more effective one [28]. We concord with this view and in our new model building environment we provide implicit user feedback in the form of simulation results which are meant to induce reflection and self-explanation.
4.8. DISCUSSION AND CONCLUSIONS

By opting for a domain independent approach we take a fundamentally different approach from traditional ITS systems. Instead of focussing on the domain knowledge that the user is supposed to acquire, we focus on the processes that are expected to lead to the acquisition of that knowledge. To this end, the system does not need to maintain a full cognitive model of the domain knowledge of the learner, it only needs to infer just enough information from the steps taken by the user to support the model building process. Thus we provide tailored feedback based on knowledge about the model building process and the constraints of the model. Another feature of our approach is that the support system takes the form of an advisory system. Contrary to other approaches to supporting users, some of which were presented here, we do not want to interrupt the user in order to offer help. The user should always be the one in control and initiate a support session if needed.

In this chapter several commonly used approaches to providing support in learning environments have been analysed with the aim of identifying positive traits which may be useful in the implementation of an adequate support system for the new version of our qualitative modelling workbench. We summarise what we have learned.

- Domain Independence brings about a number of consequences which make the provision of support more demanding from an implementation point of view. For instance, the graphical language is an extra feature to be learned for which additional support must thus be provided.

- Qualitative modelling is usually not part of standard curricula and therefore support related to the QR ontology has to be made available.

- The simulation results are per se a form of feedback and induce users to reflect. Therefore, being able to run a simulation and analyse its results in one and the same environment is an essential feature of a well-built model building environment. Furthermore, having multiple forms of presenting the simulation results is desirable since this facilitates the interpretation of those results.

- Due to the high engineering effort when applied to domain-independent systems, but also due to having opted for a constructivist approach, we do not intend to implement a student model, as done in traditional ITS systems. However, we do provide tailored feedback to some extent.

- As suggested by some of the systems presented in this chapter, we believe that pedagogical agents represent a new learning paradigm. We have thus proposed a set of agents that can guide users in the process of building models. The use of different characters to handle different aspects of support has been suggested. We are herewith taking advantage of the personification and modularity features of agents.

In Chapter 5 we present the detailed design of our new model building environment as well as its support module which were implemented using the lessons learned in this chapter.