The SMISLE Environment: Learning with and Design of Integrated Simulation Learning Environments

Ton de Jong & Wouter R. van Joolingen

University of Twente
Faculty of Educational Science
and Technology
P.O.Box 217
NL-7500 AE Enschede
Tel: +31.53.893576
Fax: +31.53.356531
e-mail: jong@edte.utwente.nl
joolingen@edte.utwente.nl

Abstract

Discovery programs, such as simulations, are frequently used in instructional setting, albeit not always with larger effectiveness. The main reason for this seems to be that learners frequently lack the proficiency to fully exploit the opportunities for discovery. Therefore, support measures are likely to be helpful to sustain learners in the (scientific) discovery process. A second problem related to simulation based discovery learning is that teachers (who are the potential authors of simulation based learning) often miss the tools and the didactical knowledge to create simulations and the support for the learner that is needed. The SMISLE environment addresses both problems. First, by identifying problems that learners encounter in discovery learning and defining support measures, and second, by providing authors with tools and information that help them to create simulations embedded in a supportive environment. In the current chapter we present a description of the SMISLE environment by following two scenarios. First, we will describe a (virtual) learner who learns in a SMISLE environment, and second, we will follow a (virtual) author who creates this learning environment with the help of SMISLE.

SMISLE

This paper gives an overview of the SMISLE environment. SMISLE (System for Multimedia Simulation Learning Environments) is an authoring environment for creating simulations that are integrated with support measures for discovery learning. We will present the SMISLE environment by introducing two scenario's. One presents a learner who learns with an application that is created with SMISLE. The other follows an author who creates this application. The application we have chosen for these scenario's is an application on the physics domain of harmonic oscillation (SETCOM, Simulation Environment for Teaching a Conceptual model of Oscillatory Motion). After a first pilot test with approximately 20 students, the application is now being evaluated with a large number of students (Van Joolingen, De Jong, & Swaak, 1995). Before we present these scenario's, we will first give a short description of the domain of oscillatory motion.

Oscillatory Motion

SETCOM addresses one-dimensional oscillatory motion for the field of mechanical engineering. Oscillatory motion is a subject taught to, for example, first-year students of mechanical engineering. In the practice of mechanical engineering, the characteristics of oscillations play an important role in design, since unintended oscillations may severely affect the behaviour of systems. Example of designs in which oscillatory motion is important are shock damping devices in cars, wings of aeroplanes, loud speakers and robots, but also designing earthquake resistant buildings requires deep knowledge of oscillations.

Ton de Jong
Wouter R. van Joolingen

Three kinds of oscillatory motion are addressed in SETCOM:

- o free oscillatory motion without friction;
- o damped motion;
- o forced oscillatory motion.

The type of motion that occurs is dependent on the presence of friction and/or an external force. In the case that both are absent, the motion is free, if only friction is present, we have damped oscillation, if both are present, there is forced oscillatory motion.

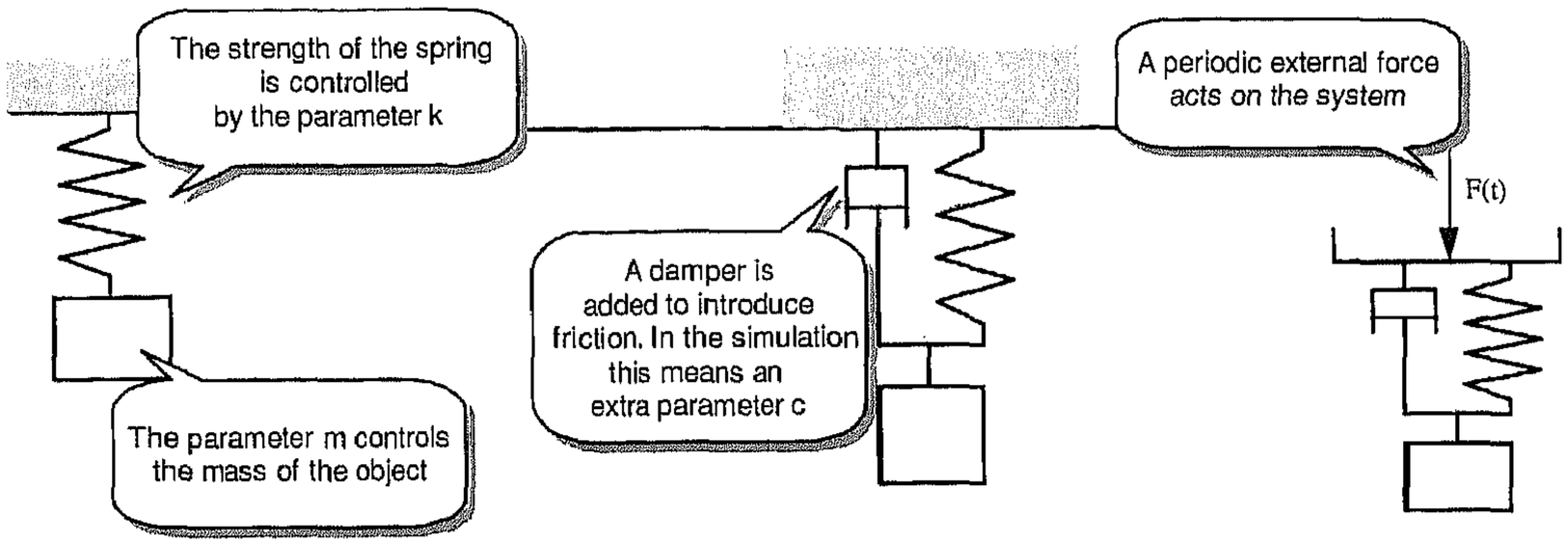

Figure 1: *Three types of oscillatory motion, illustrated by a mass suspended from a spring. From left to right: free oscillation, damped oscillation, and forced motion.*

When the frequency of the force approaches the system's eigenfrequency (free oscillation frequency), the amplitude of the system becomes very great and the phase shift approaches 90 degrees. This phenomenon is called resonance. When designing systems, it is of crucial importance to prevent situations in which resonance may occur, since these kinds of oscillations can result in unexpected system behaviour and damage.
In SETCOM, the different types of oscillatory motion are demonstrated using a simple system, a mass suspended from a string. Later a damper and an external force are added, and SETCOM ends with a complex system of an oscillating rod.

The Learning Scenario

Simulation

The central part of every SMISLE application is the simulation window. This is the window where the interface to the model is shown and where learners can enter changes to variables and observe output in the form of graphs, numerical values, animations, and the like. Figure 2 shows the simulation window of SETCOM for the model of free oscillatory motion. Here, the learner is shown a simple harmonic model, without friction and without external force. The only two input variables that can be changed are the 'spring constant' and the 'mass'. The dynamic output variables 'displacement' and 'velocity' are depicted in a (scrolling) graph and are also available as numerical values. The static output variables 'frequency of the motion' and the 'roots of the characteristic equation' are presented in a table. Input variables can be changed while the simulation is running, and effects are immediately visible. The image of the mass suspended from a spring is an animation that follows the simulation.

Ton de Jong
Wouter R. van Joolingen

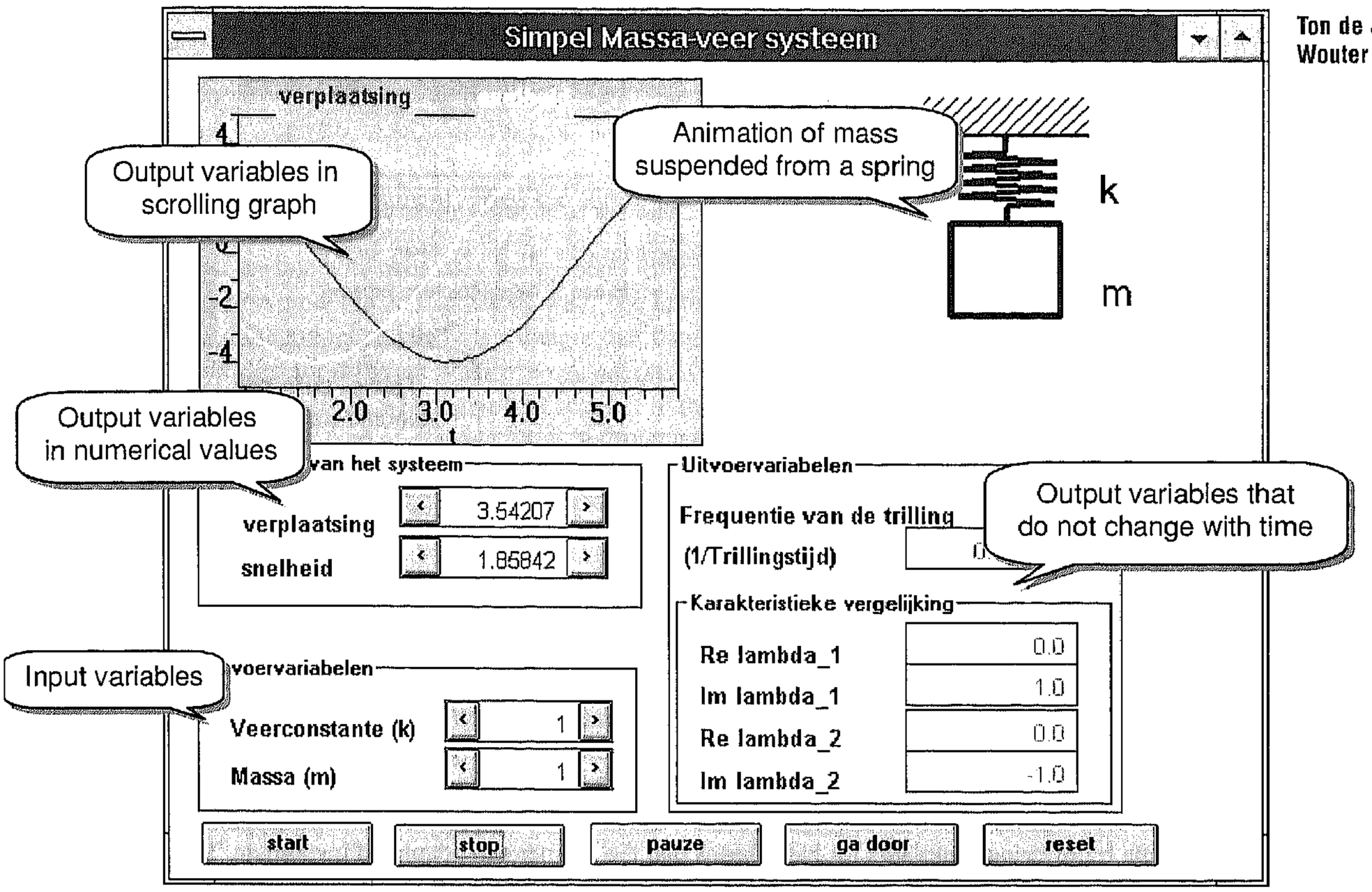

Figure 2: *Simulation window for free oscillatory motion.*

Assignments

SMISLE applications offer learners the opportunity to ask for assignments. An important aspect of discovery learning is that learners have to manage or regulate their own learning process. Assignments are designed to support the learner in regulating the learning process and intended to prevent the phenomenon of 'floundering' (Goodyear et al., 1991). This phenomenon summarises learner's behaviour that shows a learner who is 'out of control', and does not know what to do anymore.

SMISLE offers the possibility for creating different types of assignments of which the most important ones are *investigation assignments*, in which the learner is asked to investigate a specified relation in the model, *specification assignments*, in which the learner predicts the values of certain variables and *optimisation assignments*, in which the learner manipulates input variables in such a way that a certain goal is reached (see De Jong et al., 1994, for a full list). Figure 3 shows an assignment selection window. The learner can scroll through a list of available assignments and select one. In this case an assignment was selected that asked the learner to investigate the relation between the mass and the eigenfrequency of the system. For such an investigation assignments the learner should first go to the simulation window and play around until an idea has been formed. By clicking the answer button from figure 3, an answer window pops up, where, in this case, a number of alternatives is presented. The learner selects the alternative that is seen as being true and will receive feedback. This feedback can be of any kind: a new assignment, an explanation, an animation etc. The answering modes and the type of feedback is dependent of the type of assignment that is selected.

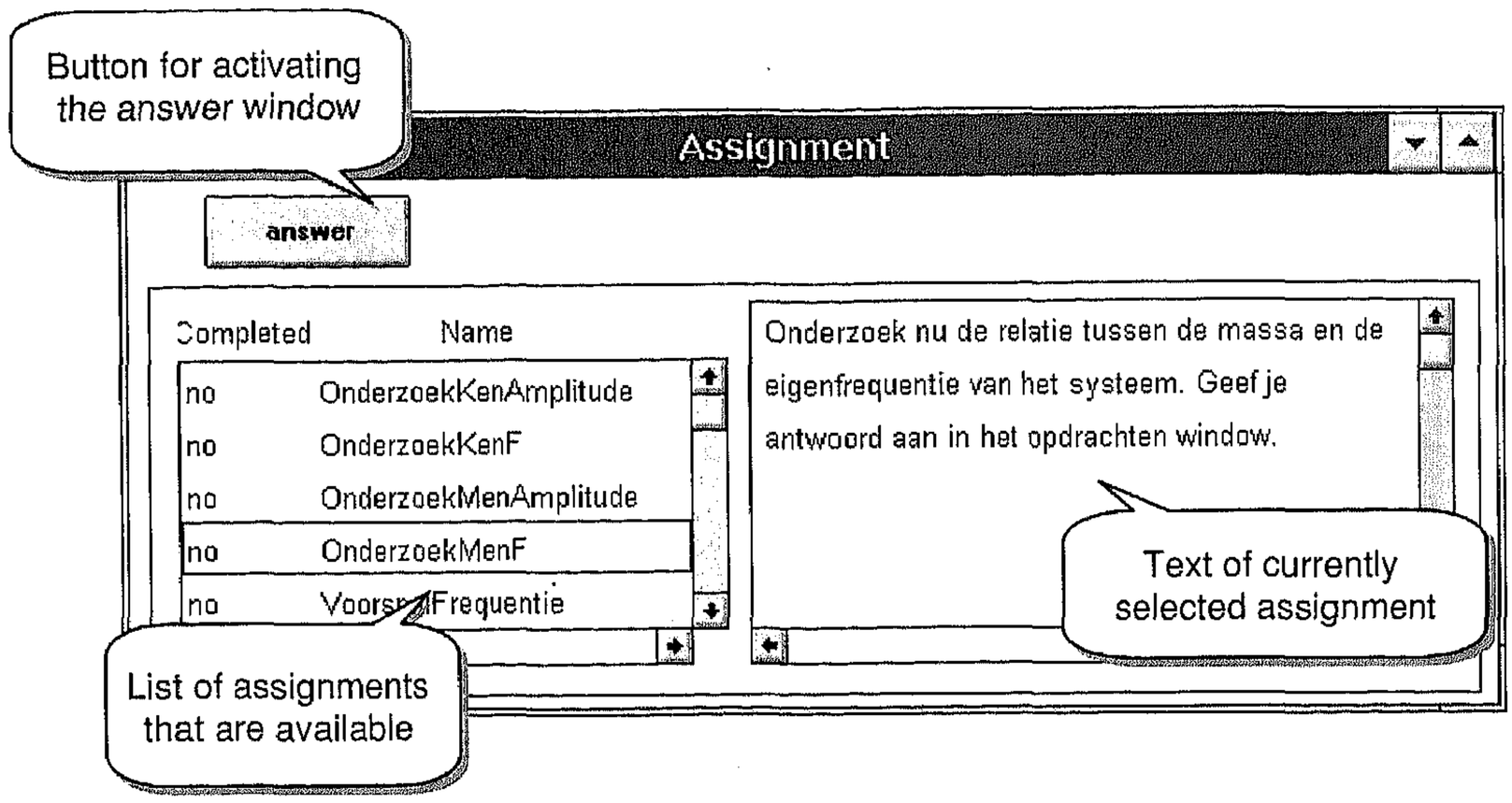

Figure 3: Assignment selection window.

Hypothesis scratchpad

Research shows that stating a hypothesis is one of the most difficult discovery processes (Klahr & Dunbar, 1988; Njoo & de Jong, 1993). SMISLE applications offer the learner support in the form of a hypothesis scratchpad, of which an example is depicted in figure 4. The idea of a hypothesis scratchpad was also used in 'Smithtown' (Shute & Glaser, 1990). A hypothesis scratchpad is a software tool that helps learners to formulate 'technically correct' hypotheses. This is done by offering templates for hypotheses that the learner can adapt and instantiate. The basic template that is offered is "if variable A relation then variable B relation". The learner can adapt this template by using logical expression that are offered, such as 'and', and 'or'. By clicking the 'then' of the basic template and clicking a button from the panel with 'logicals' a more complex template can be created. This template can then be instantiated by clicking 'variable' and selecting one of the available variables, and repeat this same action for the 'relation' parts of the templates. Functions can be added to variables. The template is automatically adapted following changes that learners make. Once a learner is satisfied with a hypothesis, the hypothesis can be saved to a list, the list can be scrolled and learners can annotate

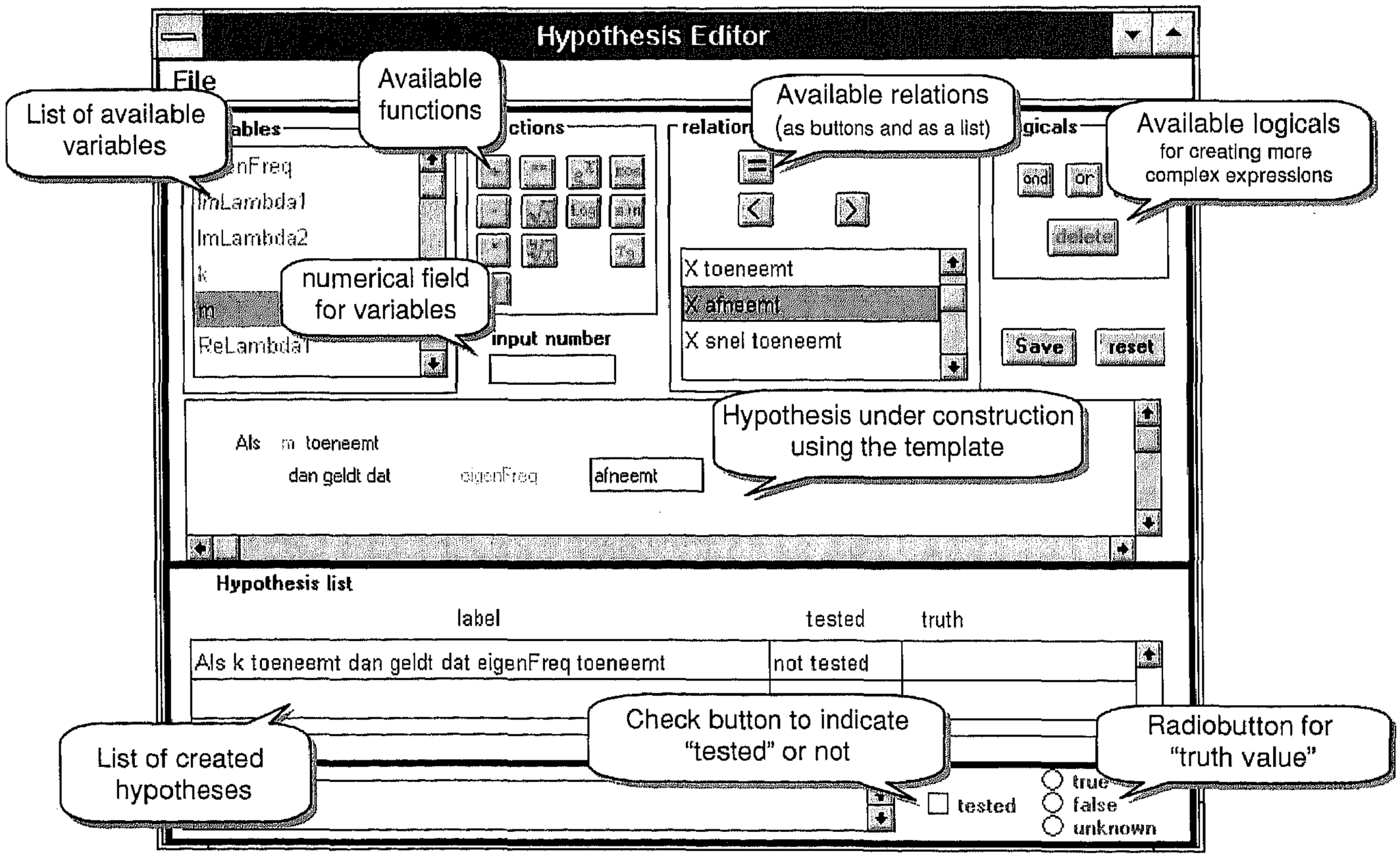

Figure 4: A hypothesis scratchpad.

whether they have tested a hypothesis, and whether they think the hypothesis is 'true', 'false', or yet 'unknown'. At present the hypothesis scratchpad merely functions as an annotation tool, for example, no feedback is as yet given to hypotheses from learners.

Explanation

The SMISLE environment gives the author the opportunity to add explanations to the learning environment. Explanations can be used for different purposes, for example for providing the learner on-line with necessary background information, or to present information from the simulation model in an expository way. The latter is useful when learners fail to discover some parts of the model and there is a danger that they will get stuck. The SMISLE environment offers possibilities for different types of explanations. One important type of explanation is the 'modulatory explanation', which gives an account of the causal relation(s) between two or more variables. This type of explanation is automatically generated using the formalism from the model in a SMISLE application. At the moment, this type of explanation can be given, but still in a quite formal representation. Other types of automatically generated explanations are: structural explanations (presenting a structural tree of variables in the model), and attributional explanations (presenting attributes of a variable). For a more complete list see De Jong et al. (1994). In addition to these automatically generated explanations authors can create explanations themselves. Figure 5 gives an example of an explanation created by an author. In this case it explains a variable: ImLambda1. In SETCOM this type of explanation only contains text and formulae, in other applications created with SMISLE extended graphics, animations, and sound are included.

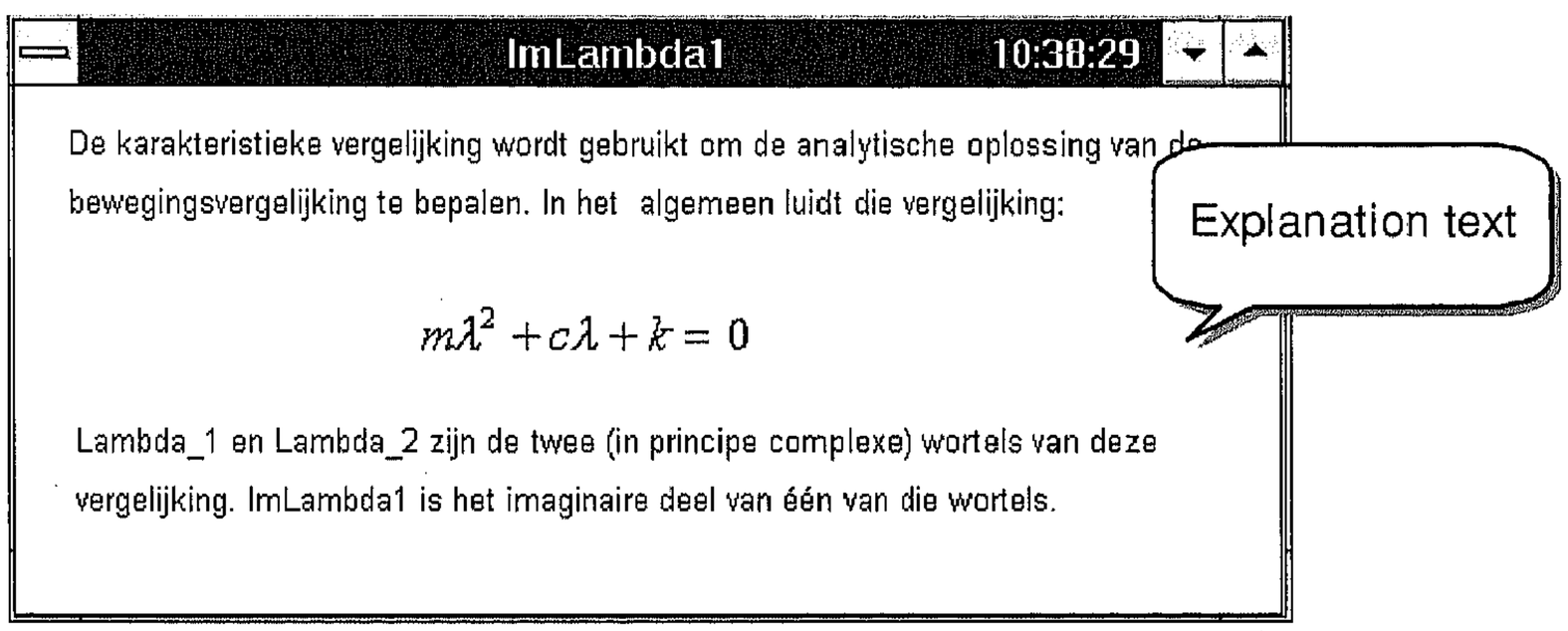

Figure 5: *An example of an 'explanation' from SETCOM.*

Model progression

The final discovery support measure that is facilitated by the SMISLE environment is model progression. Model progression (introduced in the QUEST system by White and Frederiksen, 1990) introduces a model in gradual steps, increasing the complexity at every step. SETCOM contains four levels of model progression. SETCOM starts off with free oscillatory motion, at the second level a damper is added, at the third level an external force, and finally at the fourth level it ends with a complex model consisting of a rod supported by a spring and a damper. Figure 6 shows the simulation interface of the second model progression level.
All other measures, explanations, assignments, and the hypothesis scratchpad are related to the model progression level, which for example means that the explanations and assignments differ between levels, but also that the variables that are available at the hypothesis scratchpad change with model progression level.

Ton de Jong
Wouter R. van Joolingen

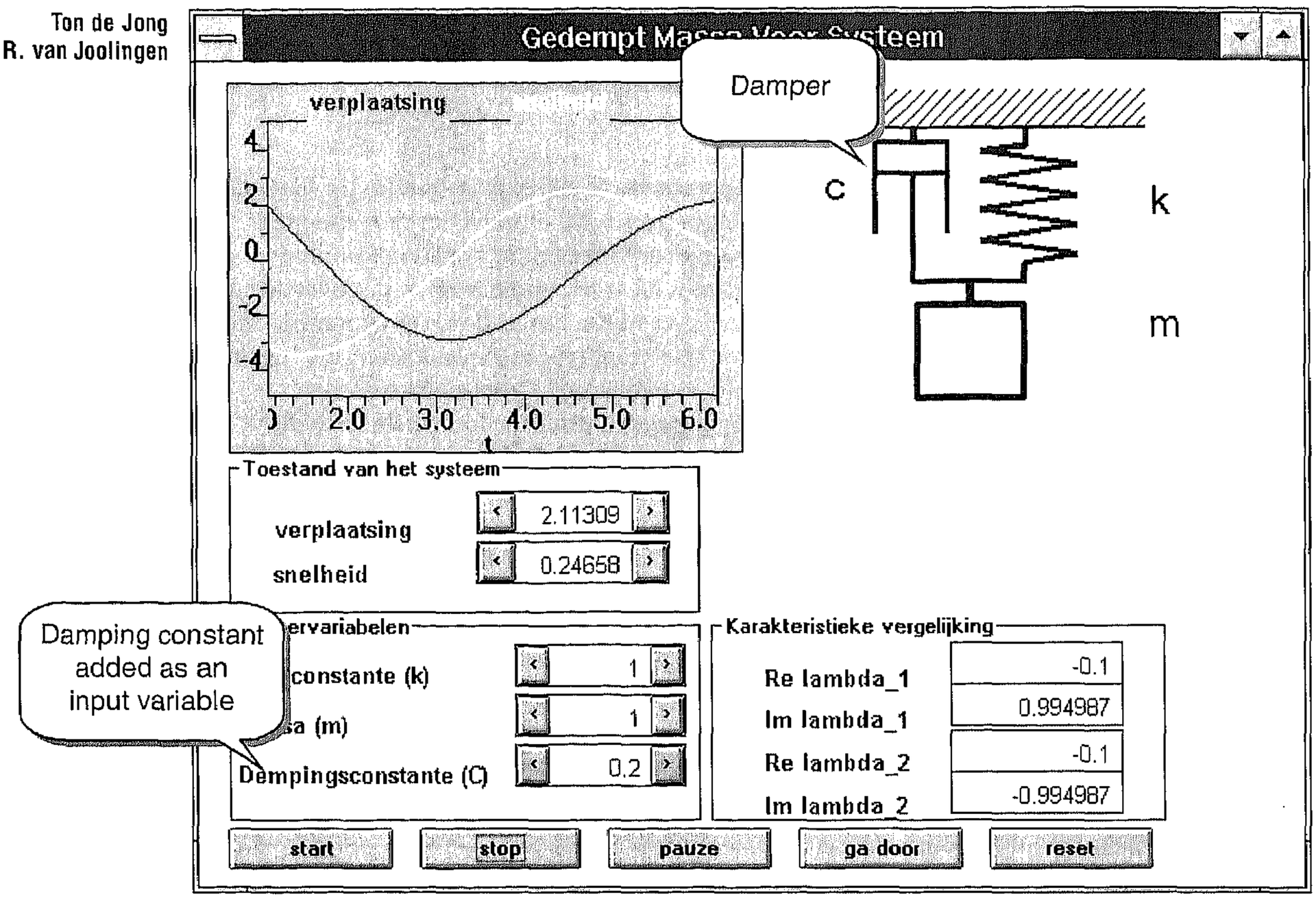

Figure 6: *Simulation window at the second model progression level. Compared to the first level (see figure 2) a damper is added, which also creates an extra input variable ('C').*

Navigating through SETCOM

The SETCOM learning environment is a complex environment. In its most elaborate form, in addition to the simulation window, there is a window that displays and describes the current model progression level, a window for assignments, a window for the hypothesis scratchpad, a window for the explanations, and a general control window. All these windows can be present at the screen at the same time, or have separate buttons by which the learner can toggle between them.

In a typical session SETCOM is started at the first model progression level that contains a model of simple harmonic motion. At this point the learner can start playing with the simulation. Alternatively, the learner can select an explanation, which will be displayed, create a hypothesis on the hypothesis scratchpad, or select an assignment. Assignments may modify the environment: usually a specific case is set up in the simulation, sometimes even the interface of the simulation is changed. If the learner has chosen an assignment, an answer attempt can be given. On success or failure the learner receives feedback in the form of explanations, or is pointed to another assignment.

As soon as an assignment is completed, the learning environment updates itself. For instance, new assignments may become available, but also new explanations, and even a new model progression. The learner sees the changes in the available instructional measures and can select a new one, including moving to a new model progression level, in which case the set of assignments, explanations, and the hypothesis scratchpad is replaced by a new one.

In SMISLE applications learners are always rather autonomous in their choices. However, the author may restrict the availability of support measures depending on the actions of the learner. This is done with the help of the 'enabling condition' editor.

The Authoring Process

Ton de Jong
Wouter R. van Joolingen

An author who wants to create a simulation-based learning environment with SMISLE is faced with three tasks:

- creating one or more models of the domain;
- creating a learner interface;
- creating instructional measures for supporting the learner in learning with the simulation.

In this section we describe these tasks in more detail in the context of the SETCOM application. and indicate how the author is supported by the SMISLE system. Our target author is a teacher who is a domain expert, but who is not necessarily a programming expert or an expert on didactics. All authoring tasks in SMISLE are based on the principle of selecting, adapting, instantiating, and connecting building blocks from predefined libraries.

Creating models of the domain

The core of any simulation-based learning environment is a model of the domain. The first task of an author who uses SMISLE is to create one or more of these models. This modelling task is the first task for an author to perform, however, later during the authoring process, the author can always modify the models that were initially created.

The SMISLE system includes a modelling tool that uses a *hierarchical functional block* formalism. This formalism is especially powerful when the system consists of distinguishable components. Each component can then be modelled separately, and connected in a simple fashion to create a complete model.

In our case of oscillatory motion, the author has to start with an analysis of the model(s) that will be used: the important components are identified and modelled separately. The author opts for using model progression as one of the instructional measures which implies that this analysis has to be made for each model that will be used. Of course, components of one level can be reused on another one. In the case of simple harmonic motion, the system consists of components that can be found directly in the SMISLE default functional block libraries: a spring, that is modelled by a capacitor, and a mass, that is modelled by an inertia. Capacitors and inertia's are part of the bondgraph formalism, which is provided in SMISLE as a part of the functional block library.

In addition to the system model, also a theoretical component associated with the system is created: the characteristic equation. This equation is necessary for understanding the exact behaviour of the system. The simulation will calculate the solutions of this equation for the learner.

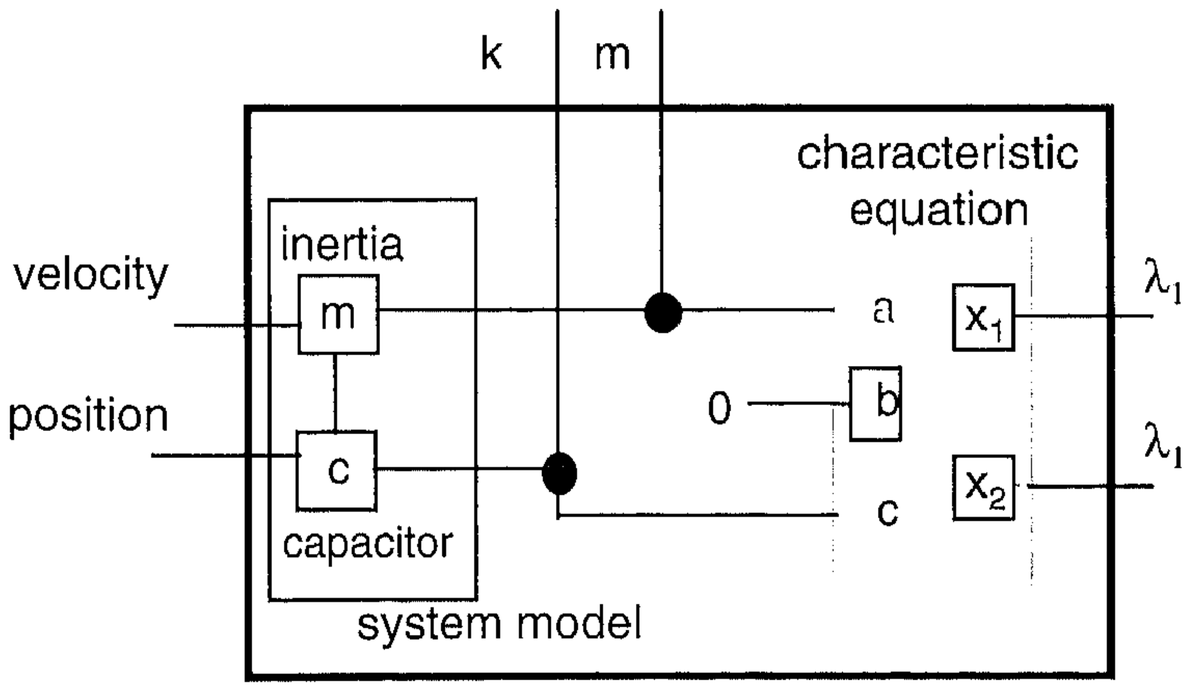

Figure 7: *The structure of the domain model: blocks are connected to form a complete model. The thick box denotes the separation between the internal structure of the block and the outside appearance.*

The resulting model of the first level of model progression is depicted in figure 7. Input variables are depicted on top of the large box, output variables of the system model are shown at the left, and of the characteristic equation at the right. The model consists of two main components: the system model and the characteristic equation. The component for the characteristic equation is modelled once and reused at all different levels of model progression, the system model is extended for the various level of model progression, to include friction, external forces, and more complex topologies of systems.

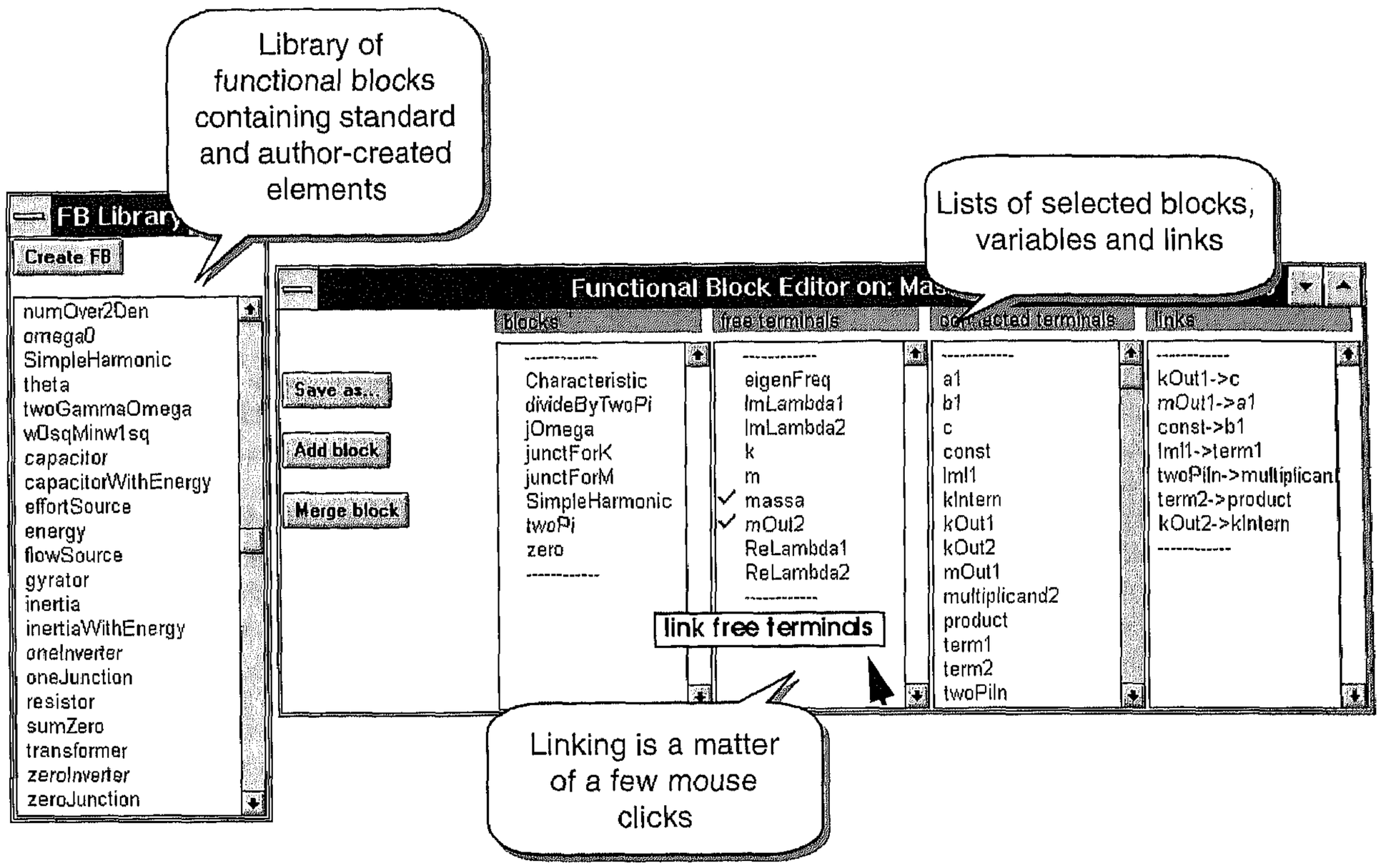

Figure 8: *The modelling tool for selecting and linking functional blocks to form a complete model.*

When the author has identified the main components of the system as given in figure 7, the SMISLE modelling tool can be used to build the model. In the SMISLE modelling tool, blocks are selected from the functional block library, and placed in the model, as is shown in Figure 8. A block is placed in the model, by selecting it in the library and pressing "add block" in the modelling tool. The input and output variables of this block then appear as "free terminals" in the model editor. The free terminals of a block can be connected to free terminals of any other block. As a block is finished, it can be saved, meaning that it is placed in the library for later use. In such a way, the model is built incrementally. The final block an author creates is the block containing the complete model. This block can be compiled, and from that moment on, the qualities of the model are known to the SMISLE authoring system, so that the author can build upon the model to create the interface and instructional model.

Creating a learner interface

For creating a learner interface to a simulation model, SMISLE offers the author an interface tool. This tool consists of a library of interface elements and a tool for placing the elements in a window and attaching them to variables in the model.
As a typical start, the author has to decide how to represent the model on the screen: just numbers, graphs, static graphics, animation, or a combination of these elements. In the example we discuss here the author chooses for the latter: graphs, numbers, and animation will all be present on the screen. The general lay-out of the screen is designed as the author 'paints' the

interface, using elements from the interface library. We will discuss the steps for creating the various elements of an interface, and the task of linking the interface to the simulation.
Creating a control, like a numerical control, directly linked to a variable is simple: the control is selected from the library and placed on the window. Then the control is attached to a variable by opening a *properties editor* for this. This properties editor allows the author to edit the interface object by filling in details of the graphical representation and linking the element to a variable in the model. Since SMISLE is an integrated environment, in the properties editor all variables from the simulation model that was created, are automatically available. Figure 9 gives an outline of this process.
Creating a graph is a two-step process. First the author specifies the nature of the graph: axes, labels, legend, using a tool much like the interface construction tool. Once the graph is ready, it can be put on the interface and attached to variables in precisely the same way as the normal controls.
A static picture can be inserted from any drawing or display tool. Pictures are put onto the SMISLE interface by grabbing the picture from a screen and placing it on the interface. Due to their static nature, there is no need for attaching pictures to variables.
Finally, animations are created by creating a series of graphic images, by grabbing each image from a screen. The animation object created in this way is attached to a variable, by linking image number to variable value. A dedicated properties editor is provided for this.
Simulation control buttons such as start, stop, pause, continue, reset, and are created by taking a button from the library and making a selecting from the available functions in the properties editor for the button.
During the editing process of the interface, the author can always directly switch to 'learner mode' for test purposes. As soon as the author selects "test", the simulation will run just like it will look during a learner session. This facilitates a quick development cycle for both the interface and the underlying model.

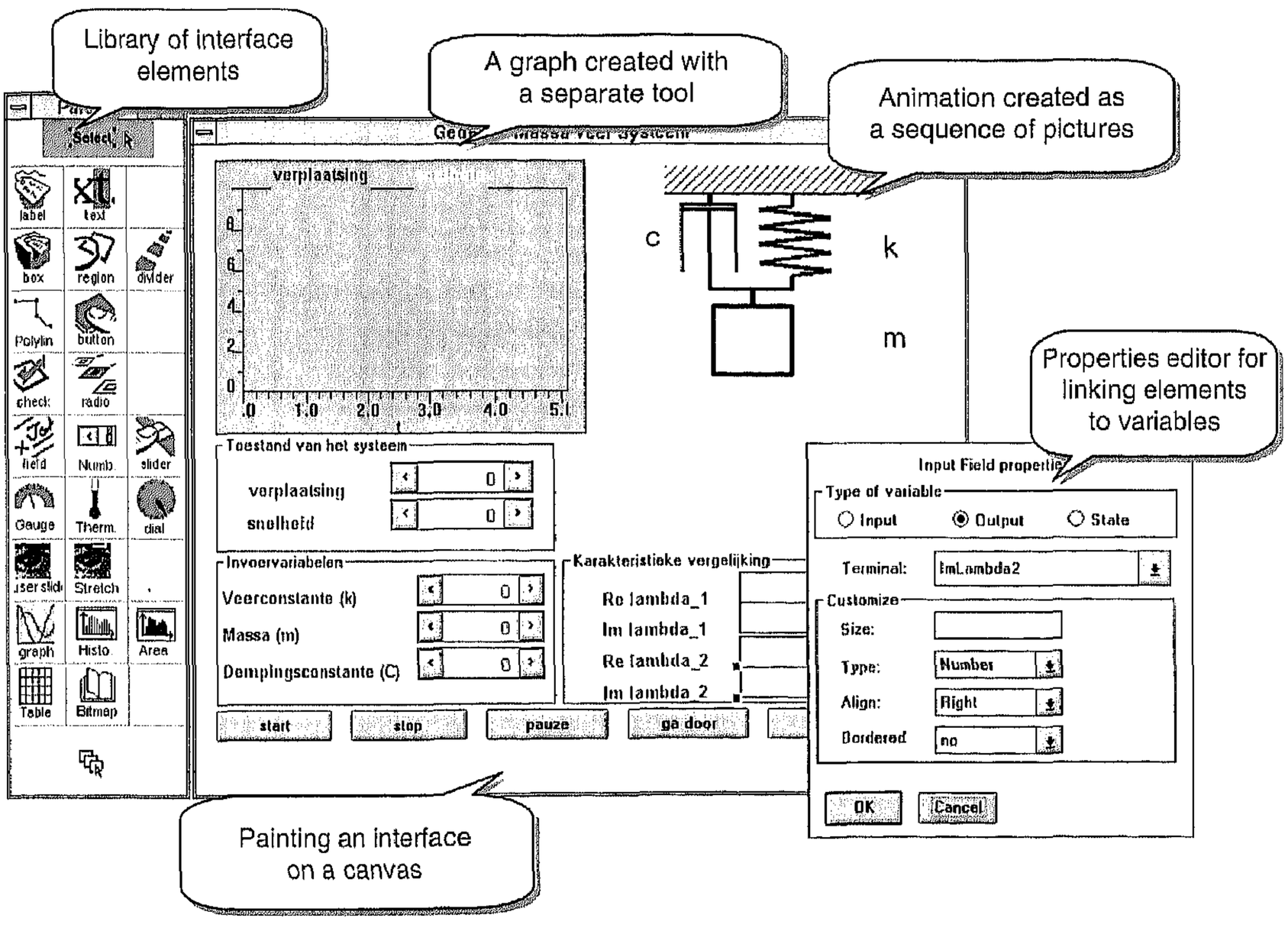

Figure 9: *Editing the interface for a model created with SMISLE.*

The instructional model

When (the first version of) the simulation and the interface have been created, the author can start with the task which is central to the process of creating a learning environment, the creation of the instructional model. From the authors point of view, the instructional model looks like a collection of instructional measures, each of which contains a small bit of instructional support. The author has to decide which instructional measures to include, these measures then have to be created, and finally, the author has to combine all measures in the learning environment.

Deciding which instructional measures to use

In creating instructional support the author first has to decide on the nature of the instructional measures to include in the learning environment. This choice is, of course, very situation dependent. In making this choice the author can call upon *author advice* included in SMISLE. This is a hypertext system, containing background information on exploratory learning, information on the instructional measures present in SMISLE, their function and tips for use, and a small expert system, which generates suggestions, based on domain and learner characteristics, entered by the author. Figure 10 shows two example screens from the author advice. Author advice can be entered in a context sensitive way from relevant other editors in SMISLE (see figure 11).

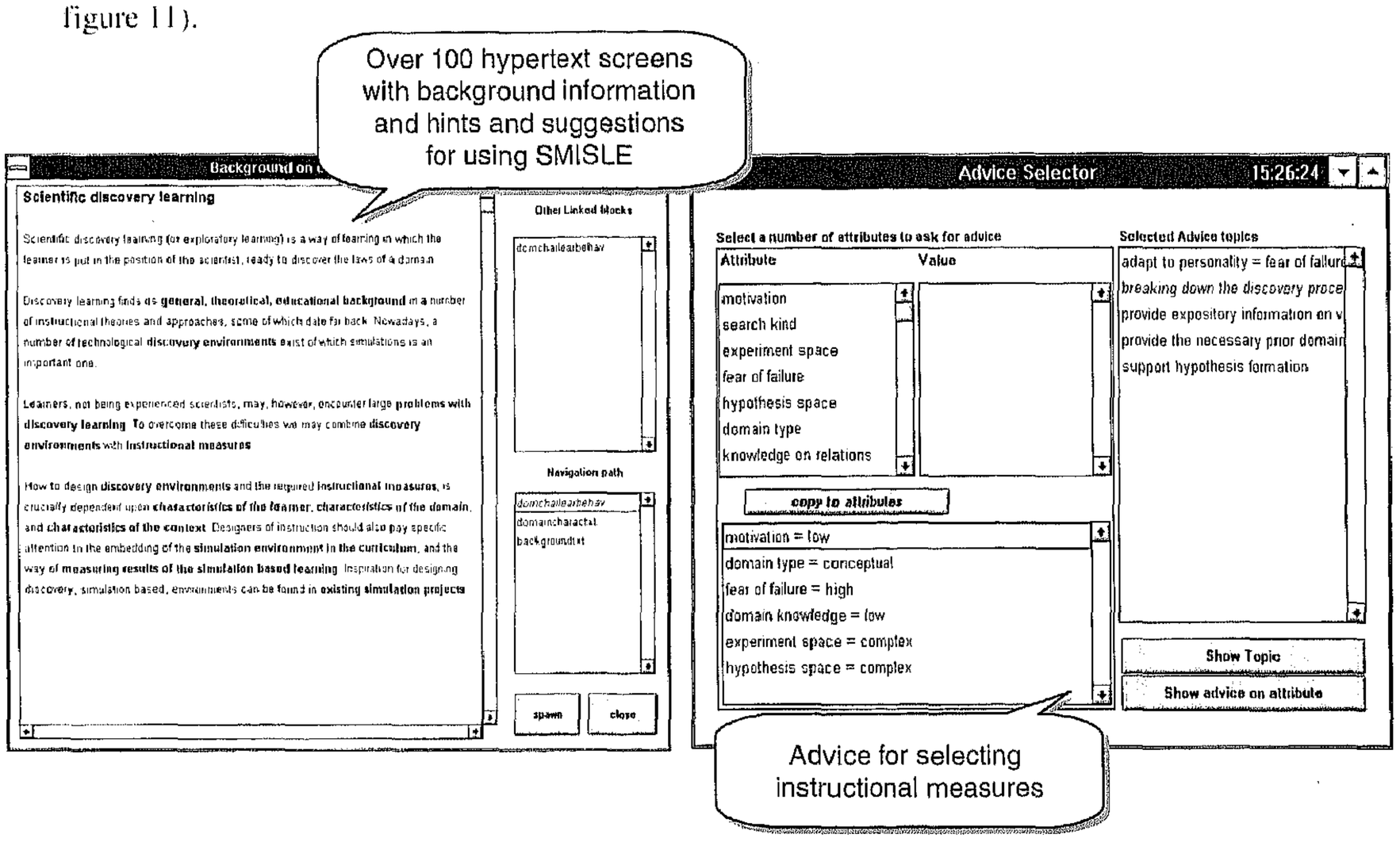

***Figure 10:** Example screens from author advice*

Creating instructional measures

In the example we discuss here, the author chooses for the following instructional measures:

- o model progression to guide the learner through a sequence of increasingly complex models;
- o a number of investigation assignments, to stimulate the learner to investigate selected aspects of the model;
- o specification assignments and optimisation assignments, to allow the learners to test themselves;
- o hypothesis scratchpads to allow the learner to state ideas about the model;
- o explanations to give some background information on variables.

Each instructional measure is created and edited separately, and can then be embedded into the structure of the instructional model. Creating an instructional measure is straightforward: the author selects a template from the library and then opens a building block editor on it to fill in the details of the particular situation. Figure 11 displays a building block editor for an optimisa-

tion assignment (all types of assignments have their own specific template). The goal of such an assignment is that the learner finds some optimum values for certain output variables by modifying the inputs of the model. In the case of the system for oscillatory motion, the author includes these assignments because they require the learner to understand the relations between input and output really well. Therefore, they can be included for learners to test themselves, and, of course, for the author or a teacher to assess if the learner really understands the subject matter. The optimisation assignment is created by setting a goal and constraints. Both are checked during run time: if the goal ("target state") is reached, the assignment is successfully completed; if a constraint is broken, the learner is sent a message and may try again. As was the case for the interface tool, the integrated character of SMISLE is illustrated by the fact that all variables from the simulation model become automatically available in the assignment editors. Finally, the author can specify feedback to success, failure, or to each answer alternative (in case the assignment provided a possibility for multiple choice answers, as is the case with for example investigation assignments).

Creating a hypothesis scratchpad is also very straightforward. The only thing an author needs to do is to give a scratchpad a name, and to indicate which elements, variables, relations etc., should appear. Again, these elements can be chosen from automatically generated lists. After selecting the relevant elements, the hypothesis scratchpad as it is depicted in figure 4, is automatically created by SMISLE.

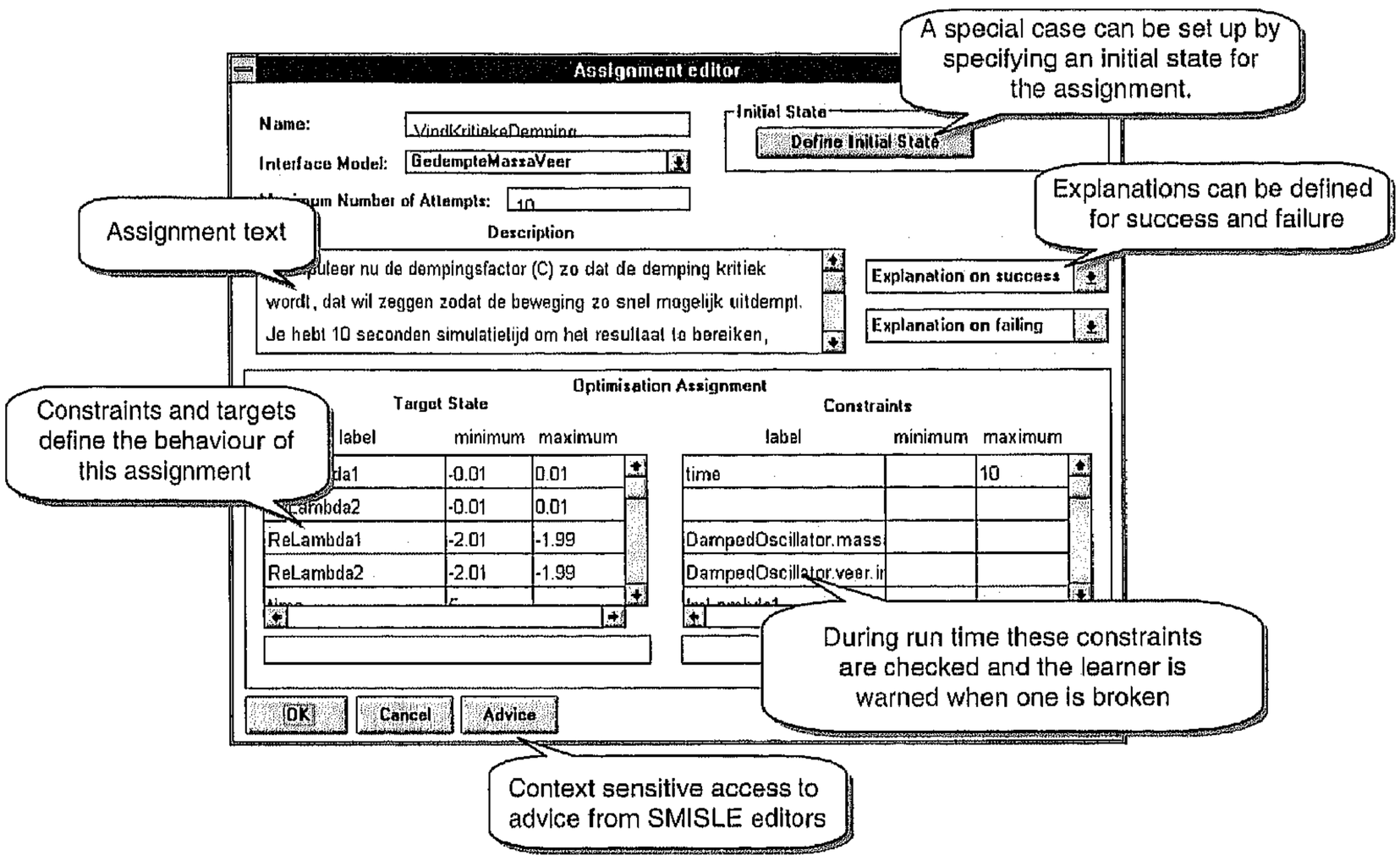

Figure 11: *A building block editor to tailor an assignment from the library to the author's situation.*

Defining the instructional route

The final task for the author is to define the complete network of instructional measures by specifying when an instructional measure will be available for a learner. Typically, an author will have created several hypothesis scratchpads, and large numbers of assignments and explanations. The availability of these specific measures is constrained by the model progression level, since specific assignments quite often only make sense at a certain model progression level. At a certain model progression level the author could decide to leave all control for selection to the learner, and make all measures available. Alternatively, authors could decide to exercise more control over the environment themselves, and to constrain the availability of measures. This is done by specifying enabling conditions for each instructional measure. An enabling condition specifies the conditions under which an instructional measure may be selected by the learner, i.e., when it appears in a list to select from. For example: "Optimisation assignment opt1 is enabled when model progression level 1 is active and Investigation assignment inv1 is success

fully completed by the learner" is a valid enabling condition. The result of this condition would be that, provided that model progression level 1 is active, as soon as the learner completes *investigation assignment inv1 is completed, opt1 will appear in the assignment window,* ready for selection by the learner. In this way, the author has complete control over the path the learner takes through the learning environment and the amount of freedom the learner has in this process. By setting weak enabling conditions, many instructional measures can be enabled at the same time allowing the learner to choose; alternatively, the author can set the enabling conditions such that at any moment there is only one enabled instructional measure. Figure 12 displays the SMISLE editor for creating and modifying enabling conditions.

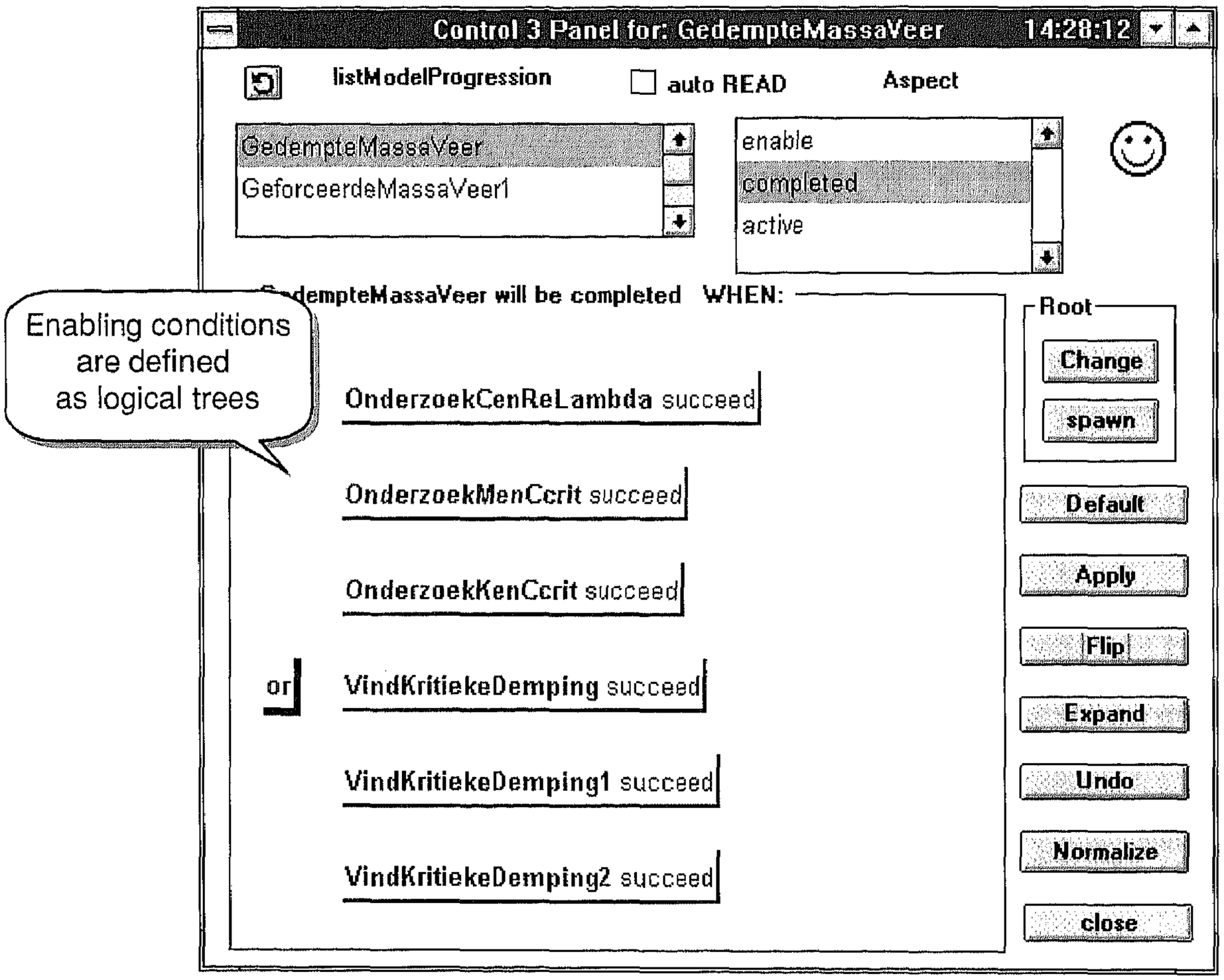

Figure 12. *A tree editor in SMISLE for specifying the enabling conditions of an instructional measure.*

Conclusion

In this chapter we have presented an overview of the SMISLE authoring environment by presenting two actors: a learner who tries to master the physics topic of oscillatory motion by working with a SMISLE application, and an author who creates this application with the use of the SMISLE environment. In the course of the SMISLE project both these actors have been subject to extensive evaluation.

Learners' behaviours and achievement are tested in a number of experimental studies with applications created with SMISLE. SETCOM (created at the University of Twente) is one of five applications currently created with SMISLE, and also, from a model perspective, the simplest one. The others are:

- Collision, on collisions (created at the University of Murcia);
- TeEl, on telegraph lines (IPN, Kiel);
- PAMSTAMP, on stamping metal sheets (Engineering Systems International, Paris);
- HPU, on a Hydrogene Purification Unit in an Ethylene plant (Marconi Simulation, Edinburgh in cooperation with Exxon Chemical).

In the evaluations an experimental paradigm is followed with different experimental groups. In order to evaluate the contribution of the different support measures in SMISLE, experimental groups differ in the availability of measures. For example, one group receives assignments, and another group learns with an environment without assignments. All learner actions are logged, cognitive load is measured during the learning process, and learners' knowledge is tapped in pre- and post-tests that are specially designed for measuring 'intuitive knowledge'. Results are not yet available, but preliminary analysis point to higher achievements with environments that contain SMISLE designed support measures.

The authoring process was evaluated during the course of the project in two ways. First, authors who created a full SMISLE application were interrogated on their experiences, and second, specific subtasks (such as creating instructional support) were intensively studied in experimental setting, with the use of techniques such as thinking aloud. In total 21 authors with different backgrounds, participated in the evaluation studies, and quite a few of the findings resulted in adaptation of the SMISLE authoring environment. A full report on the evaluation studies with authors can be found in Kuyper et al. (1994).

Future developments of the SMISLE environment will both concern the learner and the author part. For the learner part results of the evaluation studies will most likely point to problems learners experience that are currently not supported in SMISLE applications. A theoretical analysis and suggestions for support measures based on an extensive overview of the literature have already been listed (De Jong & Van Joolingen, 1995). For the author part, adaptations will include the possibility for including existing simulations, and, possibly, support for cooperative authoring.

References

Goodyear, P., Njoo, M., Hijne, H., & van Berkum, J.J.A. (1991). Learning processes, learner attributes and simulations. *Education & Computing*, *6*, 263-304.

van Joolingen, W.R., de Jong, T., & Swaak, J. (1995). SETCOM, an integrated simulation learning environment. Paper to be presented at the AERA 1995 conference. San Francisco (USA).

de Jong, T., van Joolingen, W., Scott, D., de Hoog, R., Lapied, L., Valent, R. (1994). SMISLE: System for Multimedia Integrated Simulation Learning Environments. In: T. de Jong & L. Sarti (Eds.) *Design and production of multimedia and simulation based learning material* (pp. 133-167). Dordrecht: Kluwer Academic Publishers.

de Jong, T., & van Joolingen, W.R. (1995). Discovery learning with computer simulations (in preparation).

Klahr, D., & Dunbar, K. (1988). Dual space search during scientific reasoning. *Cognitive Science*, *12*, 1-48.

Kuyper, M., de Hoog, R., van der Hulst, A., van Doorn, F., & Pieters, J. (1994). *Evaluation of the SMISLE workbench.* Deliverable D30-31-32 of DELTA project SMISLE D2007. University of Amsterdam/University of Twente.

Njoo, M., & de Jong, T. (1993). Exploratory learning with a computer simulation for control theory: Learning processes and instructional support. *Journal of Research in Science Teaching*, *30*, 821-844.

Shute, V.J., & Glaser, R. (1990). A large-scale evaluation of an intelligent discovery world: Smithtown. *Interactive Learning Environments*, *1*, 51-77.

White, B.Y., & Frederiksen, J.R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, *42*, 99-157.

Ton de Jong
Wouter R. van Joolingen

End Notes

Acknowledgements: The work presented here is carried out under the DELTA programme of the EC as project D2007. We gratefully acknowledge the contributions of the following, present and former, colleagues from the SMISLE project: Simon King, Daniel Delmas, Catherine Loprieno, Christian Canella, Jean-Marc Loingtier, Laurent Lapied (Framentec-Cognitech), Robert de Hoog, Anja van de Hulst, Michiel Kuiper, Bert Bredeweg (UvA), Jules Pieters, Janine Swaak, Frank van Doorn (UoT), David Scott, Ken Horne, Joe Brough (Marconi), André Alusse, Robert Valent, Jean-Louis Gregis, Christophe Janvrais (ESI), Hermann Härtel (IPN), Ernesto Martin, Jose-Miguel Zamarro, Francisco Esquembre (University of Murcia).