SimQuest, Authoring educational simulations

Wouter van Joolingen, Ton de Jong

▶ To cite this version:

Wouter van Joolingen, Ton de Jong. SimQuest, Authoring educational simulations. T, Murray, S. Ainsworth, M. Blessing. Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software, Dordrecht: Kluwer, pp.1-31, 2003. hal-00190676

HAL Id: hal-00190676
https://telearn.hal.science/hal-00190676

Submitted on 23 Nov 2007

Gepubliceerd als:

Joolingen, W.R. van, & Jong, T.de (2003). SimQuest, authoring educational simulations. *In: T. Murray, S. Blessing, S. Ainsworth: Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software.* pp. 1-31. Dordrecht: Kluwer.

WOUTER R. VAN JOOLINGEN AND TON DE JONG

SimQuest

Authoring educational simulations

Abstract. SimQuest is an authoring system for creating computer simulations embedded in an instructional environment. A typical learning environment created with SimQuest allows learners to engage in an activity of discovery learning with a simulation, supported by instructional measures from within the environment. Some of these instructional measures adapt themselves to the interaction of the learner with the simulation.
SimQuest allows the author to create various kinds of instructional support for the learner. *Assignments* provide the learner with short term goals for the learning process, *explanations* offer additional information, a *monitoring tool* allows the learner to record and replay experiments, and, moreover, the domain can be arranged according to different levels of *model progression*. Recent developments are a *hypothesis scratchpad*, *intelligent feedback*, a *modelling tool*, and facilities for *collaborative learning* in SimQuest.
SimQuest supports the full process of authoring simulation based learning environments, from creating the simulation model to defining the instructional interaction of the learning environment with the learner. Authoring a SimQuest learning environment is done in an object oriented way. The author selects templates from a library to create building blocks of the learning environment, which are then edited to match the author's requirements. The SimQuest architecture, with a *simulation context* as central element takes care of the interaction between the various building blocks. The author is supported by the contents of the library, on-line help, on-line pedagogical advice, and a wizard that can automate part of the authoring process.
SimQuest has been used to create about twenty learning environments, some of which have been evaluated extensively with students. Currently SimQuest is used by a group of teachers to create learning material for use in their classrooms. A number of simquest learning environments are now published by commercial publishers.

1 INTRODUCTION

1.1 Scientific discovery learning

Scientific discovery learning or inquiry learning is a type of learning that has gained much interest in recent years (De Jong & Van Joolingen, 1998). In (scientific) discovery learning learners are in control over the learning process, doing experiments, stating hypotheses and in such a way constructing their own knowledge of the domain at hand. This implies that for discovery learning, learners need to have access to an environment providing means for defining and performing experiments which provide the information needed to construct knowledge. Such an environment is of course can be found in nature itself. For several centuries doing experiments on natural systems has been the single most important source of information for the construction of scientific knowledge, within and outside the context of learning. Performing experiments on real systems still plays an important role in science education in schools and universities.

However, there may be specific reasons why doing experiments on natural systems is not preferred in education (Allessi & Trollip, 1985; De Jong, 1991). First of all, the natural system may not be available in the school, because the system itself or the measuring equipment required is too expensive or otherwise unavailable. Second, the natural system may be unsafe to use in a classroom. This is for instance the case with radioactive samples, toxical chemical substances or populations of certain bacteria. Third, doing experiments may be unethical, as is the case with livestock. Fourth, the time scale of experiments on natural phenomena may not fit with the classroom situation. Some experiments take days, months or even years to complete, others happen so fast that it is hard to visualize the processes that learners should learn about. Finally, experiments may consume resources that are rare, for instance use substances or energy, making it unattractive to let learners perform a reasonable number of experiments to see the trends and effects that are interesting in the domain under study.

To overcome the obstacles mentioned above, replacing natural experiments by computer simulations may be a fruitful enterprise (De Jong, 1991; Carlsen & Andre, 1992). Simulations are computer programs containing an executable model of a natural or other system, capable of computing the behavior of the modeled system by means of dedicated algorithms and presenting the results of these computations to the user. Simulations can overcome the above drawbacks of some natural systems. Once a model is available, it can be spread and copied as much as we like. Simulations are safe to use, as accidents or other harmful events only take place in the simulated world. Use of simulations also overcomes problems of ethical nature, as no real animals or humans will be involved. Simulations can also adapt their time scale, speeding up or slowing down processes in such a way that processes take a reasonable time to observe. Finally, simulations can be performed as much as we like, taking only resources necessary to run a computer program. We do not plea for a total replacement of experiments on natural systems, doing real experiments has its own value for learning. However, simulations can offer complementary benefits.

Apart from overcoming drawbacks of natural systems, simulations have something extra to offer to instruction. Simulations can visualize processes that are invisible in natural systems, for instance, they can create graphs of quantities like energy or impulse, or state vectors of quantum systems, and allow the learner to manipulate the representations available. This allows for multiple views and multiple representations of the simulated system (e.g. Ainsworth, 1999). More than natural systems, simulations offer a wealth of possibilities to integrate different representations and animate these representations, using data of the simulated system. For instance, a simulation of a harmonic oscillator such a mass suspended from a spring can show the basic animated movement of the mass, but also graphically represent this motion in several ways, in time-based graphs or phase diagrams. The graph and the animations can be built simultaneously showing the interrelation between the two representations.

Simulations can also increase the instructional value of a system, because there is more control over the kinds of events that occur and the frequency with which they occur. For instance, a in a driving simulation, accidents and near accidents can be simulated and explored, which would be impossible with a real system. Also events that can occur but are rare, such as entering a busy motorway, can be induced and practiced as often as is deemed necessary.

Notwithstanding all these alleged advantages of learning with simulations we see that learners have considerable difficulties with simulation based discovery learning. In de Jong and van Joolingen (1998) we have provided an extensive overview of these problems and indicated how simulations can be extended to help prevent or overcome these problems. Basically, we stated that simulations offer anchor points to integrate discovery learning with instructional support. Simulations encapsulate an explicit model of the simulated system, which can be used to interpret actions by the learner in order to provide tailored instruction. Also, instructional interventions, such as presenting exercises, or explaining observed phenomena in the simulation can benefit from interacting with the simulation model. Exercises can set the simulation in a predefined state and monitor the learner's behavior while carrying out the exercise. Explanations can be displayed whenever a state in the simulation occurs demanding for an explanation. In this way a simulation embedded in instructional interventions is created. These kinds of interventions are more difficult if not impossible to create with natural systems, even if the system is observed and controlled through a computer interface.

1.2 Authoring simulations for learning

Listing the features of computer simulations, and their use for discovery learning, one sees that they can (and actually do) provide a useful instrument in instruction. However, creating computer simulations embedded in instruction is a task that is not within reach of most teachers. Creating instructional simulations takes a number of steps, each requiring specific skills for its successful completion. In order to create an instructional computer simulation, one must

- create a *simulation model*. The model driving the simulation must be specified in a dedicated language, either a programming language or a special simulation language. For completing this part one must be an expert in the domain simulated and have knowledge of the target language. In the case of a general purpose programming language one must also create the algorithm to simulate the model;
- create a *learner interface* to the simulation. The simulation results must be visually represented to the learner, using graphics representing domain elements, graphs, numbers, etc. In order to be able to do this one must know how to create interfaces with available tools, and be able to create the domain specific graphics in the interface. Also, knowledge of proper user interface design is very useful here;
- create an *instructional design* of the environment. The learning environment should be perceived from the side of the learner. What will be the activities that will be performed? What is their goal? What information is needed and at what moment for the learner, apart from the output generated by the simulation model? For this step, one must have knowledge of instructional design;

- create *instructional interventions*. Exercises, explanations, and other kinds of instructional interventions should be part of a good instructional simulation. Part of the information needed for a good exercise can of course be presented off-line, but in order to make full use of the simulation interaction, instructional interventions should be able to interact with the simulation. This step requires both instructional design skills, as well as domain knowledge, especially didactical knowledge, such as knowing frequently occurring misconceptions;
- *integrate the parts of the environment* to a complete system. For the learner, the instructional simulation should appear as an integrated coherent system. This means that all parts need to co-operate and that the interactions between simulation model, learner interface and all instructional interventions should be smooth and stable. This step requires programming skills in order to create an efficient and transparent environment.

This list of tasks involved in creating simulation-based learning environments for discovery learning shows, on the one hand, that most teachers lack expertise and time for creating these environments themselves. On the other hand, off-the-shelf simulations often do not match the requirements of a specific teacher.

1.3 The SimQuest approach

SimQuest was created to serve teachers and learners involved in discovery learning. SimQuest is an authoring system dedicated to simulations for discovery learning. It has the following two goals:

- To provide *learners* with supportive environments for discovery learning, in the form of simulations, instructional measures, and cognitive tools directed at scaffolding the processes of discovery leaning.
- To provide *authors* with a flexible tool for creating simulation-based discovery learning environments, containing both technical and conceptual support for the authoring process.

SimQuest was planned as an open system for the design and implementation of simulation-based learning environments for discovery learning. The SimQuest system is an object-oriented system, meaning that a large number of predefined objects are present which can be used to compose a learning environment. Object types include: simulation models, interface elements, instructional measures, and test elements. The object orientation of SimQuest means that each element in this library acts according to a specific interface protocol, making it possible to extend the library with relatively little extra effort. The protocols include functionality for editing, copying, and linking elements. These design characteristics allow for the following characteristics of SimQuest:

- *Openness*. SimQuest is an open system, allowing additional instructional measures, simulation widgets, or modeling components to be easily added as long as they respond to the protocols defined in the documented specification. This, coupled with the notion of libraries and their specific interfaces, provide the authors with a powerful, customizable environment. SimQuest has the ability to interface not only with simulations created using the tools provided in SimQuest but also to external simulations.
- *Flexibility*. There is no strict order of work defined for the author. For instance, the author can start with building a simulation interface and progress with the simulation model in an iterative manner. Using this structure, the author can also co-operate with other authors and delegate parts of the work to specialists, for instance creating a simulation model.

These two principles allow that authors can follow their preferred way of working in creating learning environments, and that authors can tailor their work to their specific needs. In the next sections we describe SimQuest from the viewpoints of learners and authors, and we conclude with a description of the SimQuest architecture.

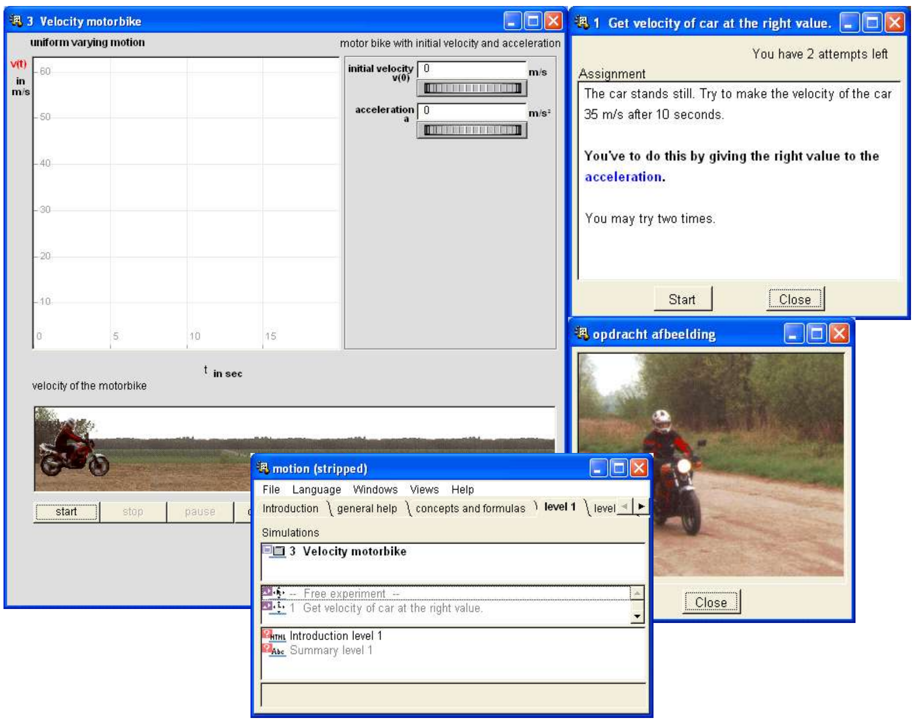
2 LEARNING WITH SimQuest

Though SimQuest learning environments (these are simulation environments created with the SimQuest authoring system) may differ considerably with respect to the domains for which they were created, all learning environment share a common structure. The nature of the activities that the learner can undertake in each of the learning environments is basically the same. The current section describes these activities and presents the tools that SimQuest offers.

2.1 Basic learning activities

The basic learning goal of any discovery-learning environment, including those created by SimQuest is to construct knowledge of the investigated domain. This does not necessarily imply that the learner must learn to know the model underlying the simulation in all detail, but the goal is to understand the principles of the domain that account for the observed behavior and/or the effects of actions performed within the domain.

In order to discover the properties of the domain and to build understanding, the learner needs to perform a number of learning processes. Several authors (Njoo & de Jong, 1993; Klahr & Dunbar, 1988; Van Joolingen & de Jong, 1997; de Jong & van Joolingen, 1998) have investigated the nature of these learning processes involved in scientific discovery learning. For instance, the learning processes identified by Njoo & de Jong (1993) include orientation, hypothesis formation, experiment design, prediction, and data analysis, as well as planning and monitoring. The activities that the learner undertakes in the learning environment should facilitate these learning processes in order to result in effective learning. SimQuest offers tools that enable and support these activities. We discuss the main components of a SimQuest learning environment and show how these components support the different learning processes. These main components are a computer simulation and instructional measures of different kinds. Figure 1 shows an example of a learning environment created with SimQuest. The learning environment is composed of one or more windows to watch and control the simulation, windows representing *instructional measures* to support the discovery learning processes and a *learner view* for the learner to control the environment.



You have 2 attempts left

Assignment

The car stands still. Try to make the velocity of the car 35 m/s after 10 seconds.

You've to do this by giving the right value to the acceleration.

You may try two times.

Figure 1. An example of a SimQuest learning environment. Shown are the learner view (bottom) providing control over all instructional measures, a simulation interface (right), and an assignment window (top right) as example of an instructional measure.

Learners working in a SimQuest learning environment will have to divide their attention between the simulation and the instructional measures. At any time the learner sees the simulation interface(s), a list of instructional measures in the learner view that can be opened, as well as windows representing active instructional measures. The learner can be involved in any of the following activities:

- Free interaction with the simulation;
- Interaction with an instructional measure;
- Interaction with the simulation within the context of an instructional measure;
- Planning and selecting which interaction with the simulation or instructional measure to perform next.

The first three of these activities can be associated with the basic process of discovery or the transformative learning processes; the fourth activity is concerned with planning and monitoring the learning process (Njoo & de Jong, 1993).

2.1.1 Operating the computer simulation

The computer simulation is present as a computational model of some system, for instance motion of a vehicle as is shown Figure 1. The computer simulation is operated by the learner through one or more *simulation interfaces*. With the interface, the learner can identify variables, modify variables, control the simulation (for instance starting and stopping it), and watch the change of variable values under influence of changes of other variables or of the progress of time.

The simulation interfaces provide the main view on the domain. All actions that the learner performs on the simulation take place through these interfaces. The interfaces, therefore, determine the view the learner will take on the domain, and hence play a role in orientation learning processes. The presence of variables as well as their position on the interface helps the learner in orientation, but also adds a bias to this process. Also, the simulation interfaces provide the main source of information and data that cater for learning processes that generate or use domain information, such as experiment design and data analysis.

The computer simulation can be operated in the context of free experimentation, or within the context of a specific instructional measure. In the latter case the instructional measure may constrain the control the learner has over the simulation interface.

2.1.2 Accessing instructional support

Instructional support is accessed through the learner view on the application. A window shows which *instructional measures* are available to the learner at a given point in time. The learner may then choose to activate one of the instructional measures.

The learner view also provides access to the *model progression* in the SimQuest application. The model progression is defined by the author as a series of models representing aspects of the domain, in the spirit of model progression as introduced by White and Frederiksen (1990). Model progressions can yield sequences of models going from simple to complex, showing different parts of a system or representing different views on the same model. Model progression is visually represented by different pages in the learner view (labeled “level 1”, level 2”,.., in Figure 1), where each page represents a progression level. Each progression page displays the simulation interfaces and instructional measures that are available on the corresponding level. Model progression can be seen as an instructional measure in itself, as it helps the learner to structure the domain, make a planning and monitor progress in learning.

The learner view provides the learner with information on the status of instructional measures: which measures are available, which are open, which have been opened and were completed and also which are not available but will be during the session.

2.2 Instructional measures

Instructional support to the learner is provided by *instructional measures*. Instructional measures represent different kinds of intervention in the learning process. Currently available instructional measures include assignments (exercises that the learner has to perform), explanations, and several supportive tools for specific learning processes.

2.2.1 Assignments

Assignments basically present support for regulative learning processes, by providing learners with short-term goals in the learning progress. The type of goal offered by the assignment differs with the assignment type. Assignments may also foster transformative processes, for example by suggesting the learner hypotheses for testing. Whenever the learner activates an assignment, it takes over some of the control over the learning environment: it *presents* itself to the learner, using text, sound or graphics, optionally it enables or disables access to variables in the simulation (blocking or hiding elements on all simulation interfaces) and defines one or more initial states, as starting points for achieving the goals presented in the assignments. For instance, an assignment that asks the learner to investigate the properties of a specific situation may set the stage for this by defining the initial states of the variables. Some assignments can offer multiple initial states with the goal of asking the learner to compare different situations.

SimQuest offers several types of assignments:

- *Do It/Do Them assignments* present their goal as such to the learner. This means that any goal can be given to the learner, but that no implicit support on achieving this goal is given from within the environment. The two versions (do it/do them) differ in the number of initial states that can be offered.
- *Investigation/explicitation assignments* ask the learner to inquire the relation between two or more variables. The assignments offer a range of possible hypotheses that can explain the observed phenomena, from which the learner can select one or more suitable descriptions. The assignments can trigger specific feedback based on the answer(s) chosen. The assignments differ in the sense that investigation assignments offer one initial state of the simulation, putting the burden of designing experiments on the learner, whereas explicitation assignments offer multiple initial states that can be seen as predefined experiments.
- *Specification assignment*s ask the learner to predict the values of certain variables when the associated simulation reaches a state specified by the author (typically a certain time). The values predicted by the learner are allowed a deviation from the simulation's values as specified by the author in either absolute or relative terms.
- *Optimization assignments* require the learner to perform a task within given constraints and with a set target. The learner has to vary the simulation's

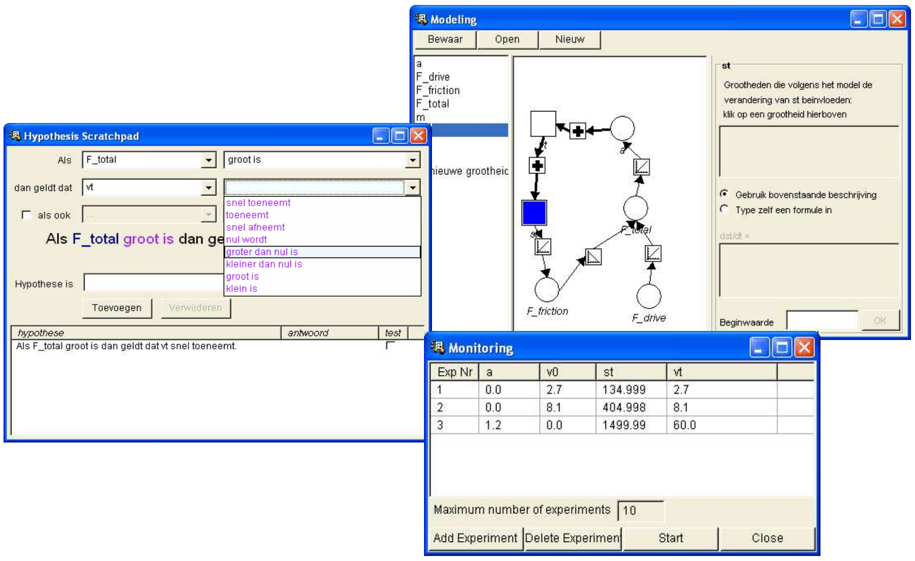
variables' values so that the constraints specified by the author are not broken and a target specified by the author is reached.

The evaluation of the answer on the assignment is performed by SimQuest. An assignment can *succeed* or *fail*, the latter meaning that the learner could not reach the goal within a specified number of attempts.

2.2.2 *Explanations*

Explanations present the learner with additional information. Explanations are classified by their content type: *text, sound, image, video, html,* or *canvas* (a collection of graphics painted on a window).

Explanations can be presented at different moments in the learning environment. First of all, they can be available on request for the learner, in the overview of available instructional measures. Second, they can be used as feedback to actions of the learner within assignments such as feedback to answer alternative for assignments that present a question in multiple-choice format, or as within optimization assignments when the learner reaches the goal or breaks a constraint. Finally, they can be used to explain simulation phenomena as they occur. In this case the explanation automatically pops up as the specific event takes place in the simulation.



Als: F_total — groot is

dan geldt dat: vt

Als F_total groot is dan ge

hypothese	antwoord	test
Als F_total groot is dan geldt dat vt snel toeneemt.		

st

Grootheden die volgens het model de verandering van st beinvloeden:
klik op een grootheid hierboven

Exp Nr	a	v0	st	vt
1	0.0	2.7	134.999	2.7
2	0.0	8.1	404.998	8.1
3	1.2	0.0	1499.99	60.0

Maximum number of experiments: 10

Figure 2. Three specific support tools in SimQuest: the hypothesis scratchpad *(left), the* modeling tool *(top right) and the* monitoring tool*, (bottom right).*

2.2.3 *Specific support tools*

In discovery learning it is important that the learner can plan and monitor the actions needed for constructing knowledge on the domain. This means monitoring the actions in *hypothesis* space (what are the ideas developed on the domain) and in *experiment* space (which experiments have been done to test these ideas. For keeping track of moves in hypothesis space SimQuest offers a *hypothesis scratchpad* and a *modeling tool*. The hypothesis scratchpad is for stating single, qualitative ideas within the learning environment, such as: when the force is great, velocity will increase rapidly. The hypothesis scratchpad was based on research by Van Joolingen and De Jong, 1991). The modeling tool offers learners the opportunity to shape hypotheses into integrated, graphical, models. Learners can specify qualitative and quantitative models that can be fed into the simulation as an alternative model to the one provided by the author. Learners can compare their model with the original model in order to test their hypotheses (Löhner, Van Joolingen & Savelsbergh, in press).

For keeping track of their moves in hypotheses, the SimQuest *monitoring tool* allows the learner to save experiments that were performed with the simulation, to sort them, and to replay them later onto allow, for instance, for comparison between experiments. The monitoring tool is especially useful in combination with investigation assignments, allowing the learner to keep track of the experiments performed. Figure 2 shows the three tools as they are presented to the learner. Each of the tools operates in close integration with the simulation and other tools present. Also, the tool design reflects the learning processes involved in the various stages of discovery learning.

2.3 Learner control vs. system control

As noted above, learners are free to choose any from the available instructional measures. This may suggest that learners are completely free in the way they progress through the learning environment. In practice there is a mixed initiative situation: the learner chooses the next step, but does so from a set of options offered by the learning environment. The size of this set determines the amount of freedom that a learner has. If there is only one choice available at any time, the learner has no freedom. If everything is available from the start, the initiative is completely with the learner.

During a session, the learner view shows these available instructional measures. Instructional measures can be activated by the learner (by selecting them in the learner view) or by the application. The learner may see that previously disabled instructional measures become enabled, instructional measures may become active without explicit request from the learner or new, previously invisible instructional measures, may appear in the learner view. This is controlled by the SimQuest instructional control system, based on author-generated specifications. These changes can be triggered by changes in state of instructional measure, for instance the successful or unsuccessful completion of an assignment, but also the occurrence of a certain event in the simulation, like a collision, passing a certain threshold, or when an otherwise interesting phenomenon takes place.

3 AUTHORING IN SimQuest

The learner is only one of the target users of SimQuest. The basic SimQuest functionality aims at authors creating SimQuest applications for their learners.

Typically, there may be three kinds of authors: professional developers of instructional software, teachers and learners, creating instructional material as part of their learning process. SIMQUEST can be used by all three; however, teachers are the primary target group.

3.1 Basic authoring principles

SIMQUEST primarily focuses on teachers who are no programmers and who have limited experience with the use of simulations for discovery learning. This means that users should not be required to write any kind of programming code, and that they should be well-supported, both on technical issues and on the design and use of simulation-based learning environments. These principles are met by the following principles that guide the design of the SIMQUEST system:

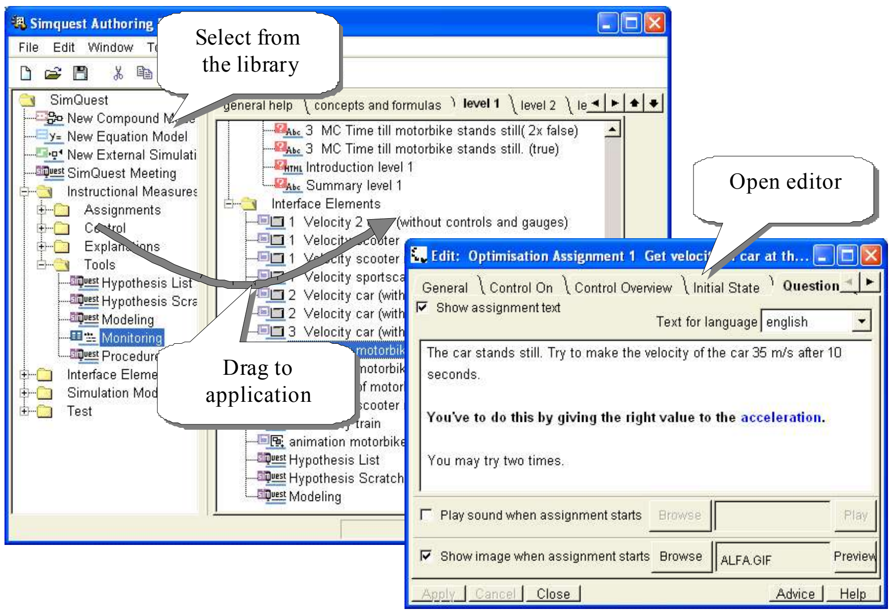
- SIMQUEST uses an object-oriented method for creating learning environments. Authoring in SIMQUEST consists of selecting, modifying, and linking pre-existing building blocks;
- SIMQUEST offers a library of building blocks for creating simulations, simulation interfaces as well as several kinds of instructional measures;
- SIMQUEST offers dedicated advice on ways of composing sensible simulation-based discovery environments;
- SIMQUEST offers context-sensitive help on the technical issues involved in creating a learning environment;
- SIMQUEST offers an intelligent wizard, taking a beginning author through all stages of the authoring process.

These five principles ensure that the authoring process is both simple for the author and powerful enough to create challenging and instructionally valid learning environments.

3.2 The authoring cycle

The authoring process is the sequence of actions that an author who works with SIMQUEST has to perform in order to build a complete learning environment. This authoring process reflects the basic principles of the SIMQUEST system. De Jong and Van Joolingen (1995) describe authoring process as: *selecting* a building block from a library, *instantiating,* and *specializing* it, and using it in a learning environment. This notion is implemented in SIMQUEST to the full extent.

In SIMQUEST an author *selects* building blocks to work with from a central resource, called the *library*. Once an author has selected a library element it can be *instantiated*, meaning that a copy of the building block is created, that can be edited by an appropriate editor. The copy is stored in the application the author is working on. *Specializing* the building block is done by opening the appropriate editor on the element. The term specializing will be replaced by ‘editing’ in the remainder of this chapter. SIMQUEST consists of a set of tools enabling this authoring process and a library containing templates for the elements on which the tools can operate. The authoring process is illustrated in Figure 3.



The car stands still. Try to make the velocity of the car 35 m/s after 10 seconds.

You've to do this by giving the right value to the acceleration.

You may try two times.

Figure 3. The authoring process in SIMQUEST, illustrated using screen shots from the actual authoring environment. Dragging a template to the application instantiates it into a building block. The author uses the editor to specializes the building block to his or her need.

The library consists of generic templates for simulation models, instructional measures, and simulation interfaces, as well as of specialized elements, suitable for use only in a limited range of domains. As a rule, generic templates are provided by SIMQUEST, but authors may add their own – specialized – library elements too for reuse in different applications.

3.3 Authoring building blocks

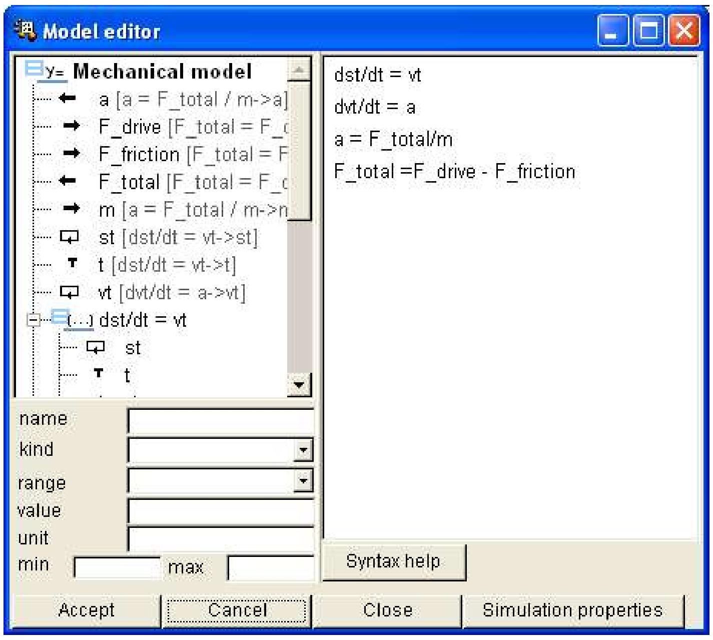
For each of the building blocks in the learning environment the basic way to author is to create it from the library of templates and to open the appropriate editor to modify the building block in order to specify its behavior in the learning environment and the interaction with other elements, for instance in executing instructional control. We discuss editing the three main categories of building blocks: simulation models, simulation interfaces and instructional measures. A separate section is dedicated to creating instructional control.

3.3.1 Creating and editing simulation models

The behavior of simulations is determined by simulation models. Simulation models specify how the values of output variables depend on the values of inputs entered by the learner and how the state of the simulation evolves over time. Simulation models are created in the same way as any other element of learning environments. However, due to the central role of simulation models in applications, the models define the structure of the application.

By creating a new model a new *model progression level* is created. Each model progression level is characterized by one underlying simulation model and the assignments and explanations for the learner that are associated with this level. For instance an author can enter models of simple harmonic oscillation, damped oscillation and oscillation controlled by an external force into one learning environment. These simulations will be visible to the learner as three different model progression levels. For each of the model progression levels, the author can create simulation interfaces and instructional measures. Model progression is a kind of instructional measure, but is technically different from the others in the sense that there is no building block in the library for defining it. It is defined by the presence of multiple models.

The author specifies the model behavior by writing equations, algebraic equations, of the form $y = a*x + b$, to compute outputs (y) from inputs (a, x, b) as well as differential equations of the form $\mathrm{d}x/\mathrm{d}t = - k * y$ to specify the dynamic evolution of the system. These equations are entered in the editor of the equation model, one of the available building blocks in the SIMQUEST library, as shown in Figure 4.



```
y= Mechanical model
   a [a = F_total / m->a]
   F_drive [F_total = F_c
   F_friction [F_total = F
   F_total [F_total = F_c
   m [a = F_total / m->n
   st [dst/dt = vt->st]
   t [dst/dt = vt->t]
   vt [dvt/dt = a->vt]
   (...) dst/dt = vt
      st
      t
```

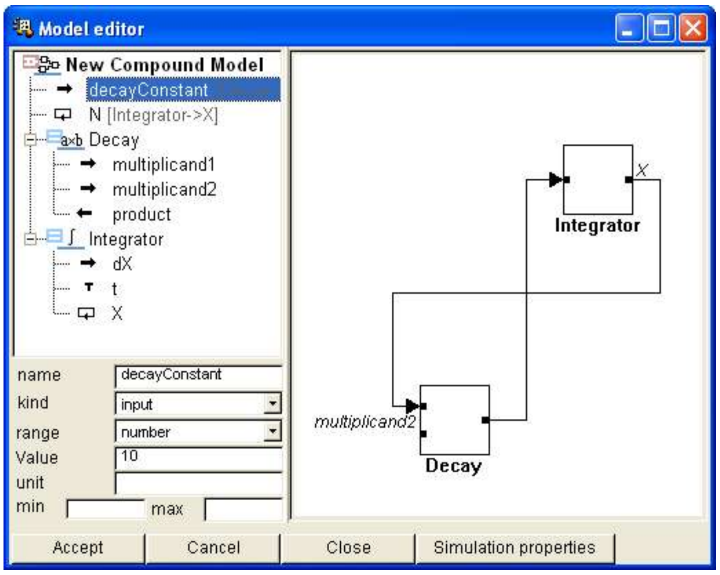
```
dst/dt = vt
dvt/dt = a
a = F_total/m
F_total =F_drive - F_friction
```

Figure 4. The SimQuest equation editor. The left displays the model structure and its variables, the right displays the equations.

Apart from the equation model, there are two other ways of creating models. One, used to form complex models consisting of large numbers of equations, is the functional block method. Smaller models are combined into larger constructs by feeding output values from one model as inputs into another. For instance a model that computes (say) a velocity of a moving body can be combined with another model that computes the body's kinetic energy from this velocity. The two models are combined into a single construct. SimQuest offers a number of models that can be used as basic material for these models. Also, authors can create their own basic model ingredients using the equation editor. The mechanism is hierarchical: functional block models can be reused in even larger structures. The functional block mechanism enables structured modeling and reuse of model components. Figure 5 displays the functional block editor.

Yet another way of creating simulation models is to use an existing, external simulation with SimQuest instruction. Currently, SimQuest provides interfaces to C++ (Borland and Microsoft), Visual Basic and Delphi. Simulations written in any of these programming languages can communicate with SimQuest. SimQuest treats these simulations in the same way as internal simulations, meaning it is possible to create SimQuest instructional measures around these simulations.

The simulation is a central resource in the learning environment. Almost any other component depends on the properties of the model and, at any moment, the current state of the simulation. The architecture of SimQuest has been designed in such a way that any component can have access to the simulation at any time, without being dependent on the internal properties of the model, for instance which type of editor was used to create it. In the section on the SimQuest architecture this is elaborated in detail.



name: decayConstant
kind: input
range: number
Value: 10
unit:
min: max:

Figure 5. The functional block model editor. On the left the model structure and variables are displayed as a tree on the right as a graph.

3.3.2 Creating and editing simulation interfaces

The simulation model can compute the behavior of the simulation; the simulation interface is used to interact with the model. It displays the values of variables, allows learners to change values of input variables and provides control over the simulation, like starting and stopping the simulation. Creating a simulation interface is done by drawing the interface on a *canvas,* as is shown in Figure 6. The components of the interface, the *widgets,* are taken from the SimQuest library and dropped on the canvas. The widget set available includes graphs, sliders, buttons, dials, gauges etc. Each of the widgets can be edited with a *properties editor* (Figure 7).

The interface editor can be used to create and edit simulation interfaces or to create and edit new, more sophisticated, widgets which can then be saved in a learning environment and then transferred to a library allowing it to be reused.

A special kind of interface widget is animations. Animations allow for a dynamic display of the evolution of the simulation state. Animations consist of a number of dynamic objects. Each object, like a rectangle or circle, has properties that can be connected to model variables. For instance, an object's position can be linked to a variable calculating the position of an object, an object's color can be linked to a variable representing, for instance, temperature.

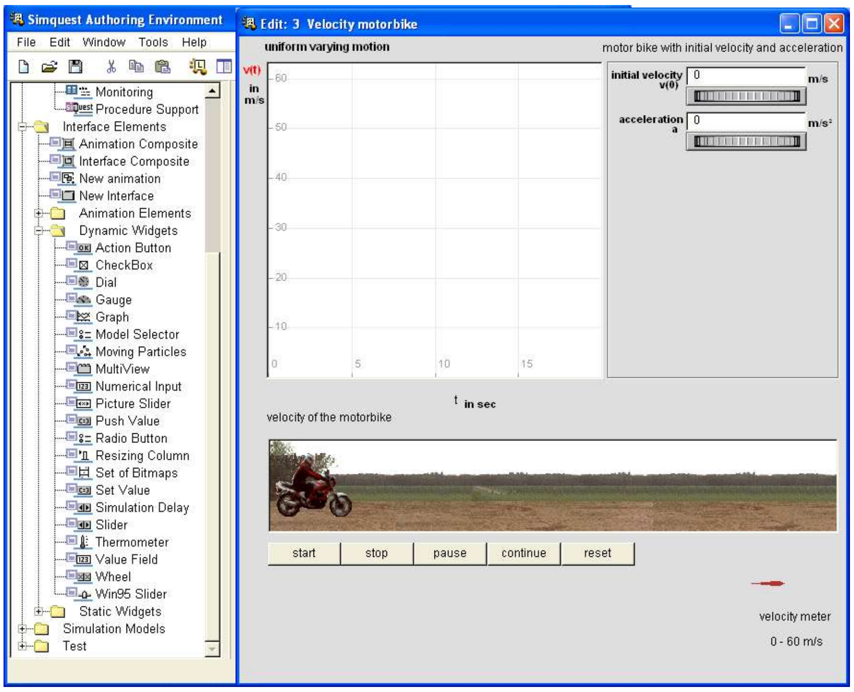
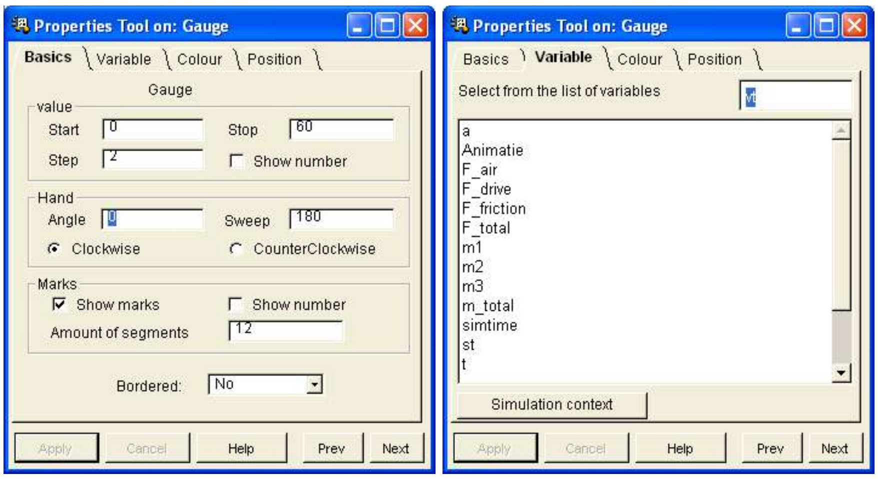


Figure 6. Editing a simulation interface. The interface is created by dragging components from the library to a "canvas". Each component can be linked to a variable in the simulation model, or with a specific action (e.g., run, pause etc.).


Properties Tool on: Gauge

Basics | Variable | Colour | Position

Gauge

value
Start: 0 Stop: 60
Step: 2 ☐ Show number

Hand
Angle: 0 Sweep: 180
[✓ Clockwise] CounterClockwise

Marks
☑ Show marks ☐ Show number
Amount of segments: 12

Bordered: No

Apply Cancel Help Prev Next

Properties Tool on: Gauge

Basics | Variable | Colour | Position

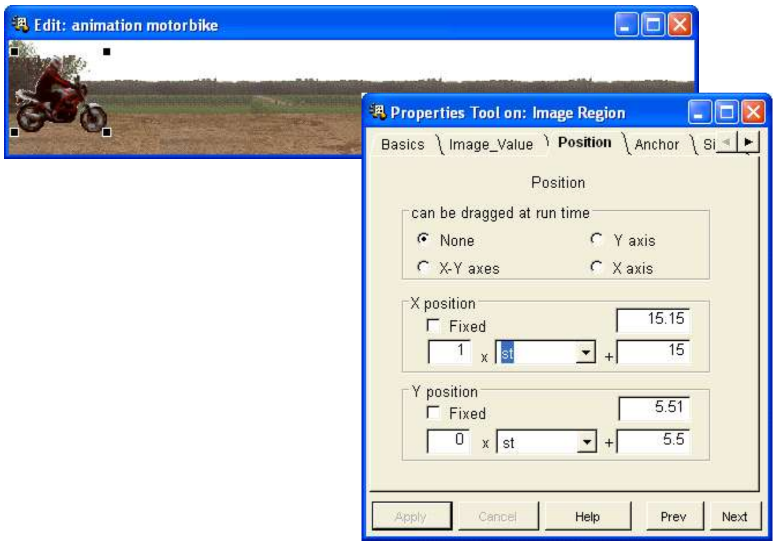
Select from the list of variables: v

a
Animatie
F_air
F_drive
F_friction
F_total
m1
m2
m3
m_total
simtime
st
t

Simulation context

Apply Cancel Help Prev Next

Figure 7. Two pages of the properties editor. One to define the variable to which the widget is linked, one to define the basic properties. The names of the variables are extracted from the model automatically, at run time the value of the variable will be displayed in the selected widget.



Position

can be dragged at run time: [✓ None] Y axis X-Y axes X axis

X position: Fixed ☐ 15.15 — 1 x st + 15

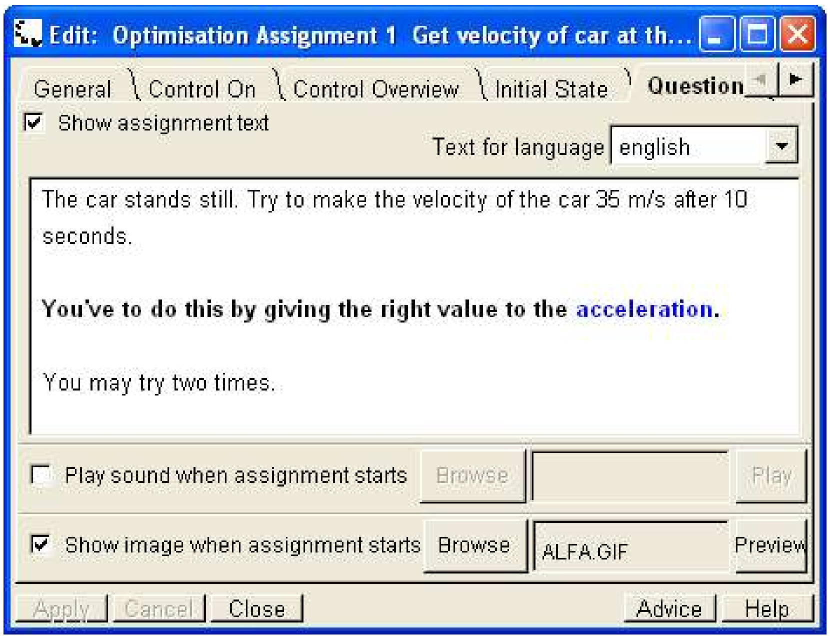
Y position: Fixed ☐ 5.51 — 0 x st + 5.5

Figure 8. Creation of animations. The way of working with this tool is very much like creating an interface. On the right side, a properties tool is shown which can be used to link attributes of the objects, like size, position, and color to variables in the simulation model.

3.3.3 Creating and editing instructional measures

Like simulation interfaces, instructional measures require close co-operation with the simulation model, and belong to the associated model progression level. Control over the simulation is needed for assignments, for instance to control access to variables or to set an initial state for the model. Once created from the library and linked to a simulation model, and the associated model progression level (by dropping the building on the relevant page) the author can open the editor for the instructional measure (Figure 9). For the case of explanations, this usually means defining the information to be displayed, like text or a media file, for assignments this means specifying the assignment goal to the learner, and determining the expected learner behavior during the assignment's activity.

Other kinds of instructional measures are edited in a similar fashion, depending on the nature of the instructional measure.



☑ Show assignment text

Text for language: english

The car stands still. Try to make the velocity of the car 35 m/s after 10 seconds.

You've to do this by giving the right value to the acceleration.

You may try two times.

☐ Play sound when assignment starts

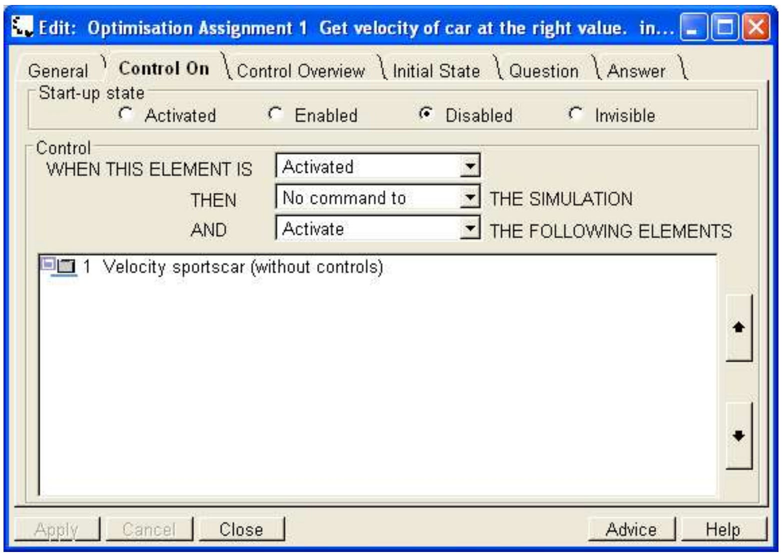
☑ Show image when assignment starts: ALFA.GIF

Figure 9. Instructional measure editor, the different pages allow the author to edit several aspects of the instructional measure.

3.4 Authoring instructional control

The SimQuest instructional control system defines which instructional measures are available to the learner at a specific moment, based on specifications by the authors. This is done with the use of *control states* that are part of the definition of any instructional measure. Different control states are: *activated*, *enabled*, *disabled*, and *invisible*. These terms apply to the kind of visibility in the learning environment. Activated means that the instructional measure is active, i.e., it displays its window so that the learner can interact with the instructional measure. Enabled and disabled mean that the instructional measure is visible on a list for the learner, where in the case of enabled the learner can select it, view the description, and activate it. Invisible means that the instructional measure is not shown to the learner. Also for each instructional measure a number of *events* are defined, for instance *activated*, *succeeded*, *failed*, and *exited* for an assignment. Events can change states of instructional measures. For instance it is possible to specify that when one assignment succeeds, a second should be activated.

For defining transitions between instructional measure states, the system uses control events generated by the instructional measures. For each instructional measure and every simulation interface, the author can define what should happen, i.e., which transitions in states of instructional measures should occur as an effect of a control event. A drag and drop interface allows the author to specify constructs such as: “when this assignment succeeds, then stop the simulation and enable assignment xyz123", as shown in Figure 10. For the purpose of control, a simulation interface is considered to be just another kind of instructional measure, meaning that they generate events on opening and closure, and that they can be activated and terminated by other control events.


Edit: Optimisation Assignment 1 Get velocity of car at the right value. in...

General | Control On | Control Overview | Initial State | Question | Answer

Start-up state
Activated Enabled [✓ Disabled] Invisible

Control
WHEN THIS ELEMENT IS Activated
THEN No command to THE SIMULATION
AND Activate THE FOLLOWING ELEMENTS

1 Velocity sportscar (without controls)

Apply Cancel Close Advice Help

Figure 10. Instructional measure editor, the page to edit instructional control.

Because the change of state of an instructional measure itself is again a control event, a chain of events may be triggered (Van Joolingen & de Jong, 1996). SimQuest offers a graphical overview over the control structure of the application.

In order to allow for more refined control over the application, SimQuest contains dedicated library elements that can generate control events other than the standard ones associated with instructional measures. The simplest of these elements is a *timer* that executes its control event when time specified by the author is reached. This element can for instance be used to specify that a learner should freely explore the simulation for 10 minutes before any assignments become available. A more complex control element is the *simulation daemon* that monitors the simulation's variables and executes its event when its constraints are broken. This can be used to attach instruction to events in the simulation. For instance an explanation may appear at the moment a critical value on one of the variables in the simulation model (e.g., a certain temperature level) is reached. This explanation may indicate why this situation is critical and what the learner can do to handle such a situation.

Finally, *composites* allow the author to group together a number of assignments in order to give them common control behavior. This behavior can specify control events attached to individual elements or subsets in the composite. This is useful if the author wants, for example, to specify complex control conditions, for example, to attach an event to the success of four out of six assignments.

4 EXTENDED SUPPORT FOR THE AUTHOR

In the SimQuest authoring process, conceptual support for the author is available in the form of templates offered in the building blocks. For example authors, as explained before, do not need to design assignments from scratch but are offered templates for the design of different types of assignment. In addition, we provide authors with explicit support in the authoring process. For designing this support, our target user was a domain expert, not necessarily an expert in programming and/or in the didactics of discovery learning. In the process of authoring the learning environment, we distinguished three main aspects: design, implementation, and methodology. For these, SimQuest offers three types of support for the author: advice, help, and a wizard (see also de Jong, et al.1999).

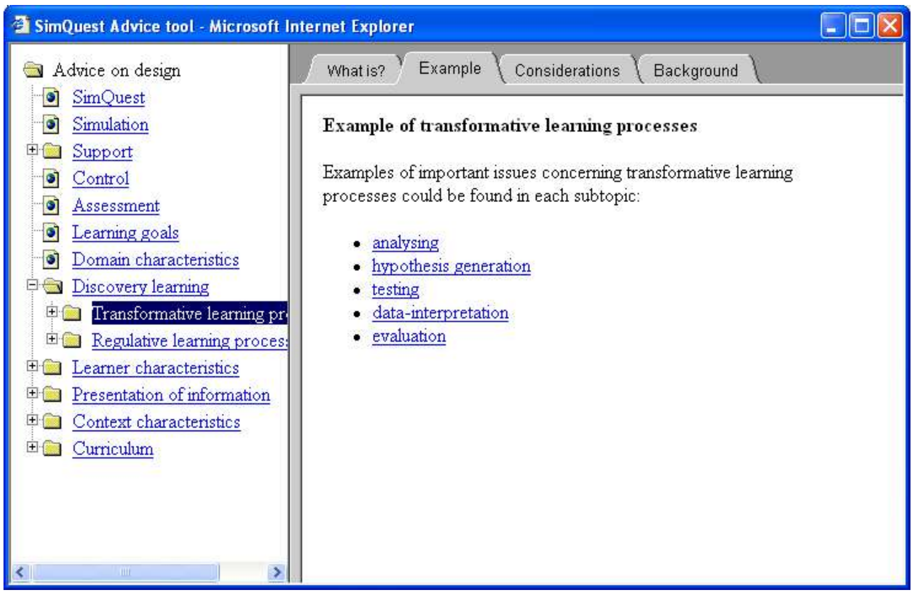
4.1 Advice

Apart from help on the technical properties of library elements, SimQuest also offers advice to authors. Authors working with SimQuest quite often have no direct experience with the design of discovery learning environments. Advice tries to support authors in this area and consists of a large hypertext like organized database of knowledge on discovery learning and its support. Advice can be accessed through the main menu, or in a context sensitive way, directly, through the editing window of an instructional measure. By pressing the advice button on an instructional measure editor, the author is taken directly to the relevant page of advice.

The advice tool consists of a two-dimensional card-tray. Different topics on which information is available are presented on the vertical dimension, and four content categories on the horizontal axis. These four content categories are:

1. "What is?" provides information of a definition or explanation of a design aspect;
2. "Example": provides (a) textual or visual example(s) of a design aspect;
3. "Considerations": the author is provided with useful considerations about a design aspect;
4. "Background" provides all kinds of background knowledge e.g., information from the literature.

Figure 11 presents an example of the advice tool interface. Through this interface and through hyperlinks within each text block authors have flexible access to the whole information system. More information on the advice tool can be found in Limbach (2000), in de Jong et al. (1999) and Limbach, de Jong, Pieters, and Rowland (1999).



Example of transformative learning processes

Examples of important issues concerning transformative learning processes could be found in each subtopic:

- analysing
- hypothesis generation
- testing
- data-interpretation
- evaluation

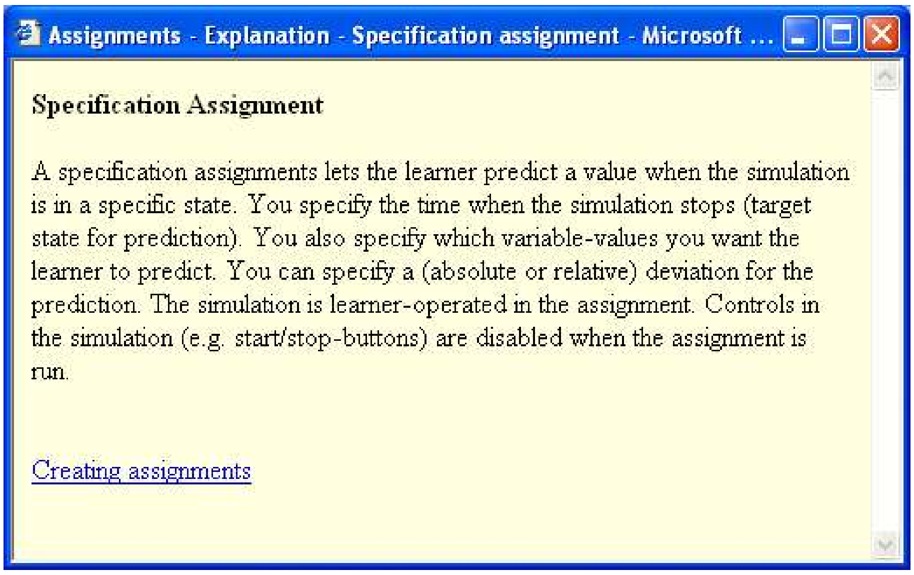
Figure 11 Example screen from the advice tool. In the example the content concerns the transformative learning process "Analyzing" and the category is "What is?"

4.2 Help

Operational support offers the user detailed steps to implement the design of the learning environment in SIMQUEST. Operational support in SIMQUEST consists of two products: online help and a manual. Both types of support follow the principles and heuristics for designing minimal documentation (Van der Meij & Carroll, 1995). This means that the help is action driven, that the tool is anchored in the task domain, that error recognition and recovery is supported, and that a principle of learning by doing is used. Figure 12 gives an example screen from the on-line manual. More information on the SIMQUEST help can be found in Gellevij (1998).

4.3 Wizard

Methodological support in SIMQUEST focuses on novice authors. By means of a wizard-like tool, these authors can generate step by step the basic structure of a SIMQUEST learning environment. This structure can later be modified and extended in the full authoring environment. Wizard support helps the author in the production process:


Assignments - Explanation - Specification assignment - Microsoft ...

Specification Assignment

A specification assignments lets the learner predict a value when the simulation is in a specific state. You specify the time when the simulation stops (target state for prediction). You also specify which variable-values you want the learner to predict. You can specify a (absolute or relative) deviation for the prediction. The simulation is learner-operated in the assignment. Controls in the simulation (e.g. start/stop-buttons) are disabled when the assignment is run.

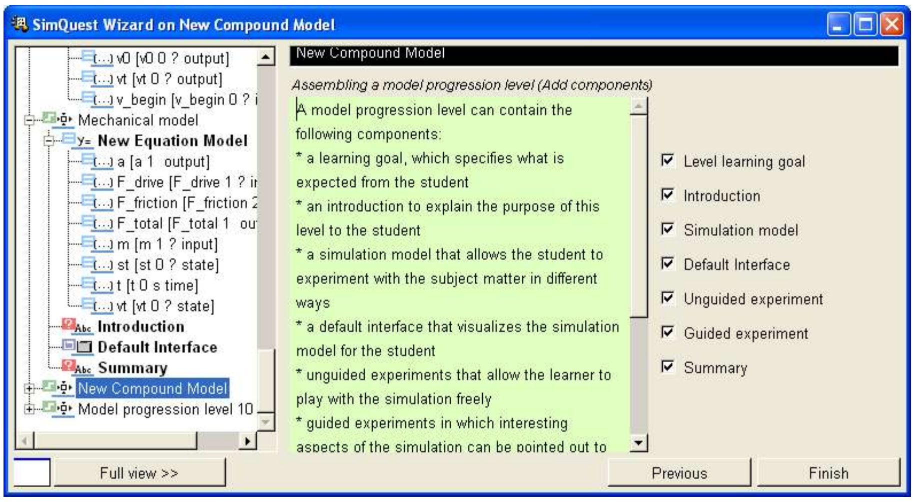
Creating assignments

Figure 12. Example window from the on-line help

- in selecting the right template from the library at the right time;
- in providing the minimally required subject matter knowledge at the right time;
- by establishing default component links, providing default computation values, and by generating default test interfaces automatically.

The wizard support also helps the author in the design process:

- in organizing the building blocks of the application in an integrated part-whole structure, that reflects the didactical decomposition of the application;
- by providing didactical decision alternatives that govern the choice of assignments.


SimQuest Wizard on New Compound Model

```
(...) v0 [v0 0 ? output]
(...) vt [vt 0 ? output]
(...) v_begin [v_begin 0 ? i
Mechanical model
  New Equation Model
    (...) a [a 1  output]
    (...) F_drive [F_drive 1 ? ir
    (...) F_friction [F_friction 2
    (...) F_total [F_total 1  ou
    (...) m [m 1 ? input]
    (...) st [st 0 ? state]
    (...) t [t 0 s time]
    (...) vt [vt 0 ? state]
  Introduction
  Default Interface
  Summary
New Compound Model
Model progression level 10
```

New Compound Model

Assembling a model progression level (Add components)

A model progression level can contain the following components:
* a learning goal, which specifies what is expected from the student
* an introduction to explain the purpose of this level to the student
* a simulation model that allows the student to experiment with the subject matter in different ways
* a default interface that visualizes the simulation model for the student
* unguided experiments that allow the learner to play with the simulation freely
* guided experiments in which interesting aspects of the simulation can be pointed out to

- [x] Level learning goal
- [x] Introduction
- [x] Simulation model
- [x] Default Interface
- [x] Unguided experiment
- [x] Guided experiment
- [x] Summary

Figure 13. Example from the SimQuest wizard

In this last respect, the SimQuest wizard is different from known wizards. It provides the author with design knowledge and also takes over part of the instructional design process. This is illustrated in Figure 13 where the wizard, based on the selected variables, provides the author with the design of assignments and interfaces for a model progression level.

5 SimQuest ARCHITECTURE

From the view of the author, creating a SimQuest application consists of creating and editing independent single building blocks. The only glue the author puts between these blocks is the control structure. All other dependencies between different building blocks, for instance, for retrieving variable values, are generated by the SimQuest system. This is made possible by the SimQuest architecture. Two aspects of the architecture are essential for a proper operation of the system: a strict separation between author-time elements and run-time elements, and a strict separation between the model and the instructional elements.

Each tool in the SimQuest authoring environment produces *specifications* of an element of a learning environment. For instance, the model editor generates a specification of the components of a simulation model in terms of variables and equations; an assignment editor will generate a specification of an assignment, etc. Once the learning environment is started, or when the author wants to test a part of the learning environment, these specifications are used to construct the actual run time elements. converted into run time objects.

Since the instructional measures interact with the simulation, a protocol has been defined for this interaction. This protocol ensures that though there may be a strong interdependency between the simulation and instructional measures at *run time*, at *author time* instructional measures and the simulation can be specified independently of each other and in any desired order. One central element in the protocol is the *simulation context.* This simulation context contains information on the variables present in the simulation model, as well as on the actions that can be performed on the simulation. This information can be created before the simulation itself. The format of information is independent of the internal model characteristics. Instructional measures and simulation interfaces can interrogate the simulation context for the names and properties of the variables present. Also they can send a request to create a new variable. There is one simulation context for every simulation model and model progression level in the learning environment.

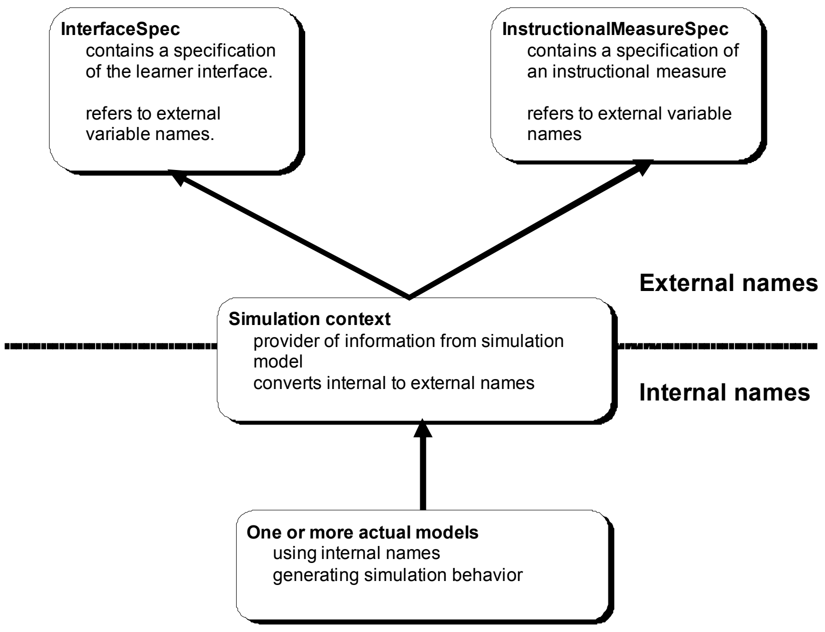

InterfaceSpec
contains a specification of the learner interface.
refers to external variable names.
InstructionalMeasureSpec
contains a specification of an instructional measure
refers to external variable names
External names
Simulation context
provider of information from simulation model
converts internal to external names
Internal names
One or more actual models
using internal names
generating simulation behavior

Figure 14. The structure of the SimQuest authoring system. Arrows indicate information flow. In a SimQuest application, any number of simulation contexts may be present. There is one for each model present in the application.

The simulation context is a platform of communication between all kinds of elements in a learning environment. Also, it can be a platform for communication between different co-operating authors. For instance, one author can create a model to suit a description stored in a simulation context, another author can use the same simulation context to design an interface. The common description stored in the simulation context ensures that, eventually, the interface and the model will co-operate. The strict separation between models and instructional measures, created by the introduction of the simulation context, ensures that the authoring process can be flexible which means that an author can use any order in creating the elemenst of the learning environment, that multiple authors can co-operate, and that elements can be reused over applications. The simulation context also allows the author to provide meaningful names for variables. Syntactical conventions in naming variables in the underlying model can be hidden from the learner. E.g., a variable oscillator1.energy.kinetic can be shown to the learner as “Kinetic energy of spring 1”.

During run time, a specialized version of the simulation context manages the interaction between the model, the simulation interface, and the instructional measures. This runnable simulation context contains a number of so-called *value holders*, one for each variable in the model. These value holders communicate the value for the variable they hold to all elements dependent of it, for instance, a graph displaying the value of a certain variable will be notified each time this value changes. Also, if from the interface a learner wants to change the value of a variable, this is done through the appropriate value holder, to which the interface is linked.

6 DISCUSSION AND CONCLUSIONS

In this contribution we described SimQuest as a system supporting authors, typically teachers or domain experts, in the process of creating simulation based learning environments, and learners in the processes of discovery learning. SimQuest is the successor of the SMISLE system (de Jong & van Joolingen, 1995; van Joolingen, & de Jong, 1996) and adds to that system flexibility, extensibility and author support. Authors are supported in the creation of challenging interactive learning environments based on computer simulations. They can create these environments without the need for programming, and can concentrate on the domain. Learners are supported in the processes of discovery by means of dedicated instructional measures that aim at scaffolding the discovery learning processes. In this sense SimQuest is both a tool and a body of knowledge on discovery learning. This knowledge is present explicitly, in the advice support for the author, and implicitly, in the design of the library and its elements.

SimQuest focuses on discovery learning of conceptual domains. This was a choice made in the design that propagates in various aspects of the system. In order to illustrate the effects of this choice, we compare SimQuest with RIDES (Munro, this volume), a tool for simulation-based learning that aims at the teaching and learning of operational skills. The first, and main, difference is found in the way the domain is modeled in both systems. SimQuest models the domain as a self standing entity describing the conceptual relations at an abstract level. A modeler in RIDES starts from the interface and adds behavior to interface elements, defined as effects of interface actions that represent the atoms of the procedure to be learned. Second, the interface in SimQuest is not seen as part of the model, but as a *view* on it. This strict separation allows for different views on the domain model, as well as views that can be adapted by the learner. Implicit in the RIDES method for specifying a model is that this separation between interface and model is not possible. Finally, the collection of instructional measures in SimQuest is aimed at supporting the processes of discovery. Typical examples are the investigation assignment and the monitoring tool. RIDES’ teaching is directed at modeling and supporting the procedure to be learned.

In its focus on discovery learning SimQuest has a strong basis in theory. The available instructional measures, as well as the way SimQuest deals with the balance between freedom and guidance for the learner, by means of the instructional control, finds is basis in theoretical work and experiments done with previous versions of the system (see for an overview de Jong & van Joolingen, 1998).

6.1 SimQuest as an authoring system

In this section we measure SimQuest along the lines indicated by Murray (this volume), and hence allow for comparison with other authoring systems. In his review the Murray mentions several classifications and measures in order to be able to compare authoring systems. We think it is most useful to position SimQuest on the level of authoring methods that is subdivided by Murray into authoring tool goals and general authoring features. This comparison allows us to position SimQuest as an *authoring system*.
Murray (this volume, p. XXX) lists five general goals for authoring tools. In SimQuest we aim to support these goals in various ways, as indicated in Table 1. In a footnote Murray (this volume) also mentions as a possible goal that the author learns something about pedagogy and instructional design. The SimQuest advice tool was designed with that purpose. The possible missing factor is that SimQuest does not *explicitly* support or enforce specific teaching strategies. The author can learn about pedagogical design with the advice module, but the author cannot pick such a strategy and use this as a template for a whole learning environment (or a significant part of it). Templates are offered on the level of instructional measures, not on their sequence. Authors have to construct the pedagogical strategy from these basic building blocks.
A second set of dimensions identified by Murray (this volume) is that of authoring features. Here the focus is on what the author can actually do with the authoring tool. Table 2 provides an overview of how several SimQuest features match against this list. As can be seen, SimQuest provides a reasonable portion of these features. Two features stand out. First, the highly modular design of SimQuest that helps the author in reusing parts of applications in others. Second, the turnaround time of editing and testing (parts of) a learning environment is short. In this way SimQuest provides authors with a flexible, supportive, working environment.

6.2 Use and evaluation of SimQuest

SimQuest has been used to create approximately twenty applications so far. The domains in which these applications were made vary from physics (collisions, electricity) via biology and chemistry, to economics and geography. The subject matters in the latter domains include a simulation to explore a model of economic development of countries in Africa. Using the SimQuest animation tool a dynamic colored map of Africa shows the consequences of manipulating parameters in this model. This range of applications shows that simulation based learning is not limited to technical and natural science domains, but can also be applied in the humanities. Important to notice is that the role of "model" may differ for various domains. An economic model has a different status than a model of physics. The latter is directed more towards explanation, the first more towards understanding of observed phenomena.

Table 1. SIMQUEST classified on the five goals for authoring tools as indicated by Murray (this volume)

Authoring tool goal according to Murray (this volume).	**SimQuest's attempt to support this goal.**
Decrease the effort for making intelligent tutors.	SimQuest tries to achieve this by reducing the effort in programming by offering standard templates, and automating the integration of modules within the system.
Decrease the skill threshold for building intelligent tutors.	SimQuest decreases the skill threshold by completely hiding the level of programming from the author. An author with a reasonable experience with general computing tools will be able to learn to work with SimQuest on the technical level. The help module and wizard are complementary measures on this point.
Help the author articulate or organize pedagogical knowledge	The overall design of the system and of the target applications helps authors to organize their pedagogical knowledge of the domain. Instructional measures are organized according to model progression levels.
Support good design principles	The SimQuest advice module provides authors with support for choosing instructional measures and instructional strategies.
Enable rapid prototyping	Every part of the design of a learning environment can be prototyped rapidly. Simulation interfaces and instructional measures can be run from within the authoring environment and be displayed in the form they will take in the learning environment. Also the sequence of instructional measures can be tested in this way.

Table 2. SIMQUEST authoring features following Murray's taxonomy.

Authoring feature (Murray, this volume)	**SimQuest Implementation**
WYSIWIG editing and rapid testing.	Present in SimQuest. Simulation interfaces can be created with a WYSIWIG editor and tested from within the editor. Instructional measures can be tested from within the authoring environment.

Flexible opportunistic design	The design of the environment can be adapted on the fly. The wizard provides a top-down design, but the author can also start with basic components and screen designs.
Visual reification	The instructional control structure can be visualized as a graph.
Content modularity and re-usability	Modularity is high. Each instructional measure, model or simulation interface operates as a separate component. Authors can save applications as libraries in order to reuse components in other applications.
Customization, extensibility and scriptability	The library function is one kind of extensibility, author created components become available as building blocks for a new application. Due to its component-based design SimQuest itself can be extended relatively simple by a knowledgeable programmer. The team that maintains SimQuest honors requests from users.
Undo	Unfortunately SimQuest does not provide an undo function.
Administrative features	SimQuest allows generating a summary description of the learning environment created for administrative purposes. Also SimQuest can keep log files of the interaction with a learner that can be summarized as a web page.
Include customizable representational formalisms	For creating models two representational formalisms are available. It is not possible for the author to introduce a new representational formalism.
Anchor usability on familiar authoring paradigms	With the event-based formalism to structure sequencing of instruction, the author can use various algorithms. The wizard offers a few more traditional and more advanced paradigms, and the author is free to add his or her own.
Facilitate design at pedagogical relevant level of abstraction	We believe that the level of "instructional measure" and its instantiations like "assignment", "explanation" are the relevant level of abstraction.

Several of the environments created have been evaluated in empirical studies. The environment *Collisions* about two colliding particles under different conditions has been used as a source material for several empirical studies (Swaak, van Joolingen, & de Jong, 1998; Swaak & de Jong, 2001; Veermans, van Joolingen & de Jong, 2000; Veermans 2003). In these studies various aspects of learning environments (configurations of model progression and assignments, intelligent feedback on experimentation behavior) were varied. An overall result of these studies is that the SimQuest instructional measures seem to be able to make a difference on the behavior of learners and in some cases also on the results on domain related knowledge tests. Other learning environments have been used to test out the modeling tool (Löhner, Van Joolingen, & Savelsbergh, in press), the hypothesis scratchpad and SimQuest's role in collaborative simulation-based learning (Gijlers & de Jong, submitted; Saab & Van Joolingen, 2003). This work is as yet more premature, as the relation between collaborative learning and simulation-based learning is still open for exploration. Veermans, van Joolingen and de Jong (2000) added a new type of explanation to the system that provided feedback on the experimental behavior of the learners. This type of explanation appeared to have a strong effect on the learner's behavior, indicating a more reflective way of working with the simulation. Veermans (2003) took this further by building in *heuristics* in the monitoring tool. The tool advises and constrains learners according to given heuristics ("vary one variable at a time" or "use canonical values for input variables"). The result of offering heuristics is that in particular weak learners profit from the structure imposed by the explicit heuristics. They are making the larger gains from pretest to posttest.

SimQuest as an authoring system has been evaluated on different aspects. Especially the measures for supporting the author in the process of creating simulation based learning environments, the help advice and wizard have been evaluated in empirical studies (Gellevij, 1998; Gellevij, van der Meij, de Jong, & Pieters, 1999; Kuyper, de Hoog & de Jong, in press; Limbach, 2000; Limbach de Jong, Pieters, & Rowland, 1999). A main lesson learned from these studies is that, though SimQuest can take away much of the burden of the authoring process, the conceptual part of authoring in itself requires considerable training for the author. So, while bringing authoring simulation based learning within the reach of a new group, SimQuest also introduces the need for extensive support for these authors. The built-in support can only partly fulfill this need, training and support by experienced users is necessary. A new approach with SimQuest is adopted by Vreman- de Olde and De Jong (submitted), who use simulation-based authoring as a learning tool for students. Learners learn by creating SimQuest assignments around a given model. By constructing the assignment they learn to understand the model.

Currently SimQuest is used by teachers at secondary schools in the Netherlands within the context of various projects and as regular part of a science method. Teachers act as authors and test their applications with their learners. Some SimQuest applications have found their way to educational practice. In middle vocational education and on high school level applications in physics and chemistry are shipped by educational publishers as part of their methods. Future developments of SimQuest are foreseen in the Co-Lab (Collaborative Laboratory) system that is currently under development. This will include dedicated support for learners collaborating on a discovery learning task, support for learners creating executable models of phenomena they observe, and integration with on-line measurements.

7 ACKNOWLEDGEMENT

SimQuest was developed in the EU-funded project SERVIVE under contract ET1020. We would like to thank the many people that contributed to the system: Andre Alusse, Miguel Celdran, Francesco Esquiembre, Thomas Gamradt, Mark Gellevij, Hannie Gijlers, Daniel Gureghian, Hermann Härtel, Robert de Hoog, Simon King, Michiel Kuyper, Renate Limbach, Simone Löhner, Ernesto Martin, Humberto Martinez, Jan van der Meij, Sven-Ole Paulsen, Jules Pieters, Elwin Savelsbergh, David Scott, Janine Swaak, Koen Veermans, Paul Weustink and José-Miguel Zamaro.

Wouter van Joolingen
Graduate School of Teaching and Learning
University of Amsterdam
Wibautstraat 2-4
1091 GM Amsterdam
The Netherlands

Ton de Jong
Faculty of Behavioral Sciences
University of Twente
P.O. box 217
7500 AE Enschede
The Netherlands

8 REFERENCES

Ainsworth, S. (1999). The functions of multiple representations. *Computers and Education, 33*, 131-152.

Alessi S.M., & Trollip, S.R. (1985). *Computer based instruction, methods and development*. Englewood Cliffs, NY: Prentice-Hall.

Carlsen, D.D., & Andre, T. (1992). Use of a microcomputer simulation and conceptual change text to overcome students preconceptions about electric circuits. *Journal of Computer-Based Instruction*, *19*, 105-109.

Gellevij, M. (1998). Operational support in SimQuest. In T. de Jong & W.R. van Joolingen (Eds.) *SimQuest: an authoring system for integrated simulation learning environments* (pp. 95-113). Internal report. Enschede: University of Twente.

Gellevij, M., van der Meij, H., de Jong, T., & Pieters, J. (1999). The effects of screen captures in manuals: A textual and two visual manuals compared. *IEEE Transactions on Professional Communication*, *42*, 77-92.

Gijlers, H. & de Jong, T. (submitted). The influence of prior knowledge on students' dialogue during a kinematics scientific discovery learning task.

Jong, T. de (1991). Learning and instruction with computer simulations. *Education & Computing, 6*, 217-230.

Jong, T. de, Limbach, R., Gellevij, M., Kuyper, M., Pieters, J., & Joolingen, W.R. van (1999). Cognitive tools to support the instructional design of simulation-based discovery learning environments: The SimQuest authoring system. In J. van den Akker, R.M. Branch, K. Gustafson N. Nieveen & T. Plomp (Eds.), *Design approaches and tools in education and training* (pp. 215-225). Dordrecht: Kluwer Academic Publishers.

Jong, T. de, Martin, E., Zamarro J-M., Esquiembre, F., Swaak, J., & Joolingen, W.R.van (1999). The integration of computer simulation and learning support; an example from the physics domain of collisions. *Journal of Research in Science Teaching*, *36*, 597-615.

Jong, T. de, & Joolingen, W.R. van (1995). The SMISLE environment: Learning with and design of integrated simulation learning environments. In P. Held & W.F. Kugemann (Eds.) *Telematics for education and training* (pp. 173-187). Amsterdam: IOS Press.

Jong, T. de, & Joolingen, W.R. van (1998). Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research*, *68*, 179-202.

Joolingen, W.R. van, & de Jong, T. (1996). Design and implementation of simulation-based discovery environments: the SMISLE solution. *Journal of Artificial Intelligence and Education*, *7*, 253-277.

Joolingen, W.R. van, & Jong, T. de (1996). Supporting the authoring process for simulation-based discovery learning. In P. Brna, A. Paiva & J. Self (Eds.), Proceedings of the European conference on Artificial Intelligence and Education (pp. 66-73). Lisbon, Portugal: Edições Colibri.

Joolingen, W.R. van, & Jong, T. de (1997). An extended dual search space model of learning with computer simulations. *Instructional Science*, *25*, 307-346.

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

Kuyper, M., de Hoog, R., & de Jong, T. (2001). Modeling and supporting the authoring process of multimedia simulation based educational software: a knowledge engineering approach. *Instructional Science, 29.* 337-259.

Limbach, R. (2000). *Supporting Instructional Designers, Towards an information system for the design of instructional discovery learning environments.* (PhD Thesis). Enschede: University of Twente.

Limbach, R., de Jong, T., Pieters, J., & Rowland, G. (1999). Supporting instructional design with an information system. In J. van den Akker, R.M. Branch, K. Gustafson N. Nieveen & T. Plomp (Eds.), *Design approaches and tools in education and training* (pp. 113-125). Dordrecht: Kluwer Academic Publishers.

Löhner S., & Joolingen, W.R. van, & Savelsbergh, E.R. (in press). The effect of external representation on constructing computer models. *Instructional Science.*

Meij, H. van der & Carroll, J. M. (1995). Principles and heuristics for designing minimalist instruction. *Technical Documentation*, *42*, 243-261.

Munro, A. (this volume). Authoring Simulation-Centered Learning Environments With Rides And Vivids.

Murray (this volume). Authoring Intelligent Tutoring Systems: An analysis of the state of the art.

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.

Saab, N., & Joolingen, W.R. van (2003). Effects of Communication Processes in Collaborative Discovery, paper to be presented at the AERA conference, Chicago, April 21-25, 2003

Swaak, J., van Joolingen, W.R., & de Jong, T. (1998). Supporting simulation-based learning; the effects of model progression and assignments on definitional and intuitive knowledge. *Learning and Instruction*, *8*, 235-253.

Swaak, J. & Jong T. de (2001). System vs. learner control in using on-line support for simulation-based discovery learning; effects on knowledge tests, interaction measures, and a subjective measure of cognitive load. *Learning Environments Research 4,*217-241.

Veermans, K. H. (2003). Intelligent support for discovery learning. PhD. Thesis. Enschede, The Netherlands: Twente University Press.

Veermans, K.H., Joolingen, W.R. van & Jong, T. de (2000). Promoting self directed learning in simulation based discovery learning environments through intelligent support. *Interactive Learning Environments*, *8*, 229-255.

Vreman- de Olde, C. & de Jong, T. (submitted). Student-generated assignments about electrical circuits in a computer simulation.

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