Instructional Science **29:** 337–359, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

Modeling and supporting the authoring process of multimedia simulation based educational software: A knowledge engineering approach

MICHIEL KUYPER[1], ROBERT DE HOOG[1,2] & TON DE JONG[2]
[1]*University of Amsterdam, Faculty of Social and Behavioral Sciences, Department of Social Science Informatics, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands (E-mail: dehoog@swi.psy.uva.nl);* [2]*University of Twente, Faculty of Educational Technology, Department of Instructional Technology, P.O. Box 217, 7500 AE Enschede, The Netherlands (E-mail: r.dehoog@edte.utwente.nl, a.j.m.dejong@edte.utwente.nl)*

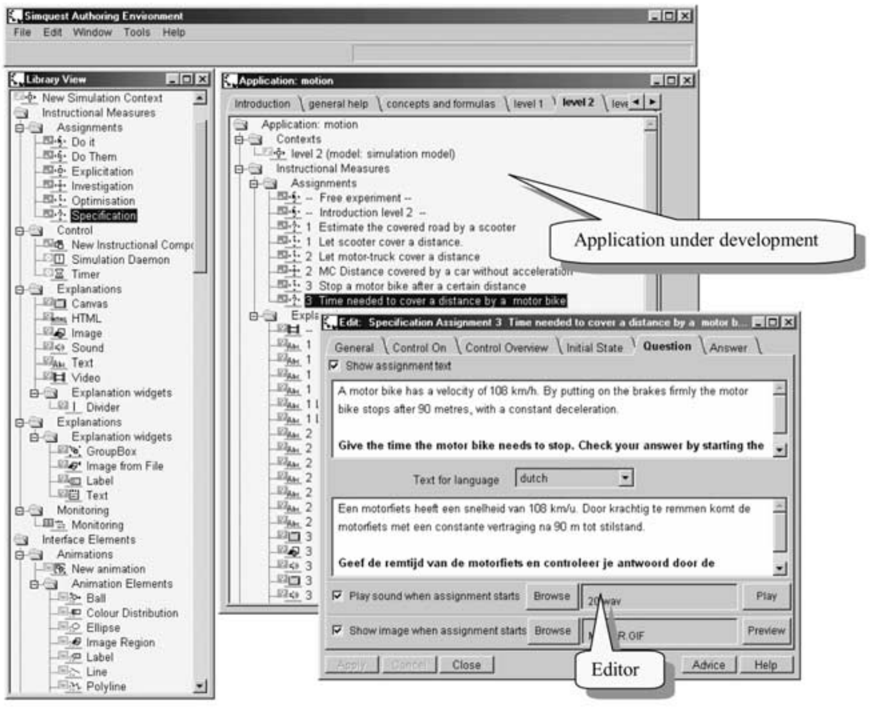
Abstract. Traditionally, support for authoring educational software focuses on the authoring process: the nature and sequence of the activities that must be performed to deliver the required product. As a consequence, the methods that are used tend to have a strong linear flavor, which resembles the classical waterfall approach. Development strategies as currently used in software engineering shift the attention from *activities* to *products* (see De Hoog et al., 1994). A general implementation of this approach can be found in the CommonKADS methodology (see Schreiber et al., 2000). The present article describes how this new development approach has influenced the design of an authoring system for multi-media simulation based educational software, the SIMQUEST authoring system.

Keywords: authoring process models, authoring products models, knowledge engineering, knowledge models, wizard

The SIMQUEST authoring system

SIMQUEST is an authoring system for designing and creating simulation-based learning environments. The unique character of SIMQUEST learning environments is that they include support for the discovery processes of the learner. This support consists of explanations, assignments, a monitoring tool, and the use of model progression. In SIMQUEST learning environments a designer has to try to find a balance between direct guidance of the learning processes and sufficient freedom for learners to regulate the learning processes themselves. The learner aspects of SIMQUEST have been extensively studied; reports can be found in De Jong et al. (1996, 1998, 1999) and Swaak et al. (1998).

Technically, a SIMQUEST learning environment consists of (a) a simulation model (or simulation models) needed to run the simulation, (b) one or more user interfaces to this model, (c) a collection of instructional supports,



A motor bike has a velocity of 108 km/h. By putting on the brakes firmly the motor bike stops after 90 metres, with a constant deceleration.

Give the time the motor bike needs to stop. Check your answer by starting the

Een motorfiets heeft een snelheid van 108 km/u. Door krachtig te remmen komt de motorfiets met een constante vertraging na 90 m tot stilstand.

Geef de remtijd van de motorfiets en controleer je antwoord door de

Figure 1. Example of the author interface of SIMQUEST, with the library (including sub-libraries to the left), and an application being built to the right.

and (d) a specification of the control flow in the learning environment (this control flow sets for example the sequence in which learners have access to specific assignments). The authoring process in SIMQUEST is an object oriented one. In SIMQUEST the author creates a learning environment by selecting building blocks from a library, instantiating and specializing them, and using them in a learning environment. The library contains pre-defined building blocks of simulation models, interfaces (and interface elements) to those models and instructional measures.

Figure 1 displays an example of a learning environment under construction. To the left the library of building blocks is displayed, to the right an application seen from the author's perspective is given (the authors may simply toggle between an author and a learner view). The library consists of predefined templates that can be used to create the instructional simulation. Examples are: assignments, small exercises for the learner; animation elements for building the interface. The author drags building blocks (e.g., animation elements or assignments) from the library to the tab sheet in the application window and uses editors on the building blocks to specialize them. The tabsheets in the application part in Figure 1 list all the elements (model, interface, assignments, explanations etc.) at model level 2 of this application. Elements are organized according to model level (where models can, for example, increase in complexity). After having selected building blocks from the library these building blocks have to be adapted to the current application (this is the specialization), which means setting preferred properties and filling in the domain content. Figure 1 shows, as an example, the editor on a so-called 'specification assignment'. For creating such an assignment the author only 'walks' through all the tab-sheets of an editor and fills in the relevant data. The general design of the assignment is incorporated in the editor. All different types of assignment in SIMQUEST have their own dedicated editor. Further information on the SIMQUEST environment can be found in Van Joolingen et al. (1997) and De Jong et al. (1998).

The SIMQUEST authoring system has been subject to an extensive evaluation with authors. Throughout its development, pilot authors have been using intermediate versions of SIMQUEST and provided the development team with bugs, questions, and feedback. From the initial author evaluation studies (see Kuyper, 1998; Kuyper et al., 1995) it was concluded that the gap between subject matter knowledge and the primitives of the authoring tool was too large. Authors not thoroughly familiar with SIMQUEST were not able to bridge this gap. The solution the authors have chosen is based on the notion of *intermediate products* between the author's knowledge and the SIMQUEST authoring primitives. These intermediate products are represented by means of so-called *knowledge models*. As a result of introducing knowledge models, the SIMQUEST environment was enhanced by a support mechanism (a 'wizard') which acts as a performance support system for developing the instructional simulation by the 'implicit' building of these intermediate products. Figure 2 shows the specific framework for supporting authoring in SIMQUEST.

Figure 2 can be rephrased in terms of two main concepts:

1. *The authoring process*: the activities performed by the author to achieve the authoring goals
2. *The authoring products*: the models needed to bridge the gap between the authoring environment and the authoring domain.

The next section is devoted to a characterization of the authoring process following principles from knowledge engineering. The authoring products are briefly described in the section on modeling the authoring process. The mapping of process and product on the SIMQUEST wizard is explained in the section on mapping models on support. The paper concludes with some observations and recommendations.

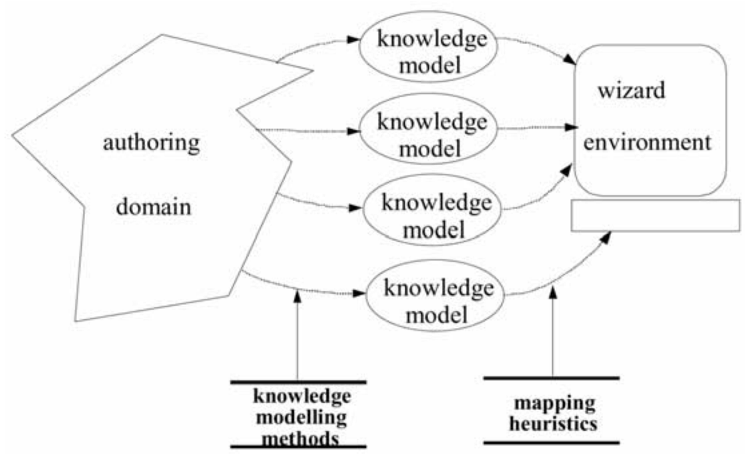


Figure 2. Framework for supporting authoring in SIMQUEST.

Modeling the authoring process

The perspective chosen for modeling the authoring process is to view it as a *knowledge intensive* task, that is, consisting of reasoning steps that require inferences to be made about the domain under consideration.

The characteristic feature of SIMQUEST is the availability of elementary *building blocks* for modeling, interfaces, and instructional support (see the previous section). This authoring process based on building blocks can be described by using from the CommonKADS methodology pre-defined generic tasks,[1] for which basic models have been developed and tested (see Schreiber at al., 2000; Breuker & Van de Velde, 1994). The authoring task in SIMQUEST is a case of design in which the elements from which the design is built are given. This type of task is called a *configuration task.*

Top and Akkermans (1994) describe a configuration task for computer-aided physical engineering. In this task they see the design process as consisting of three main steps: specification of requirements starting from the design objectives, building-block based construction of the application on the basis of requirements, and the assessment of the application in light of the objectives. This is an iterative process, which they call *evolutionary modeling*. There is a clear influence from the construction process back to the specification process; the design process can be seen as gradual refinement of requirements as the structure of solutions emerges. Figure 3 depicts an adapted version of Top and Akkermans' model, for authoring in SIMQUEST.

The elongated ovals represent reasoning steps the author has to perform. The rectangles represent the results of these reasoning steps and the components available as input for these reasoning steps. The open rectangles cover the

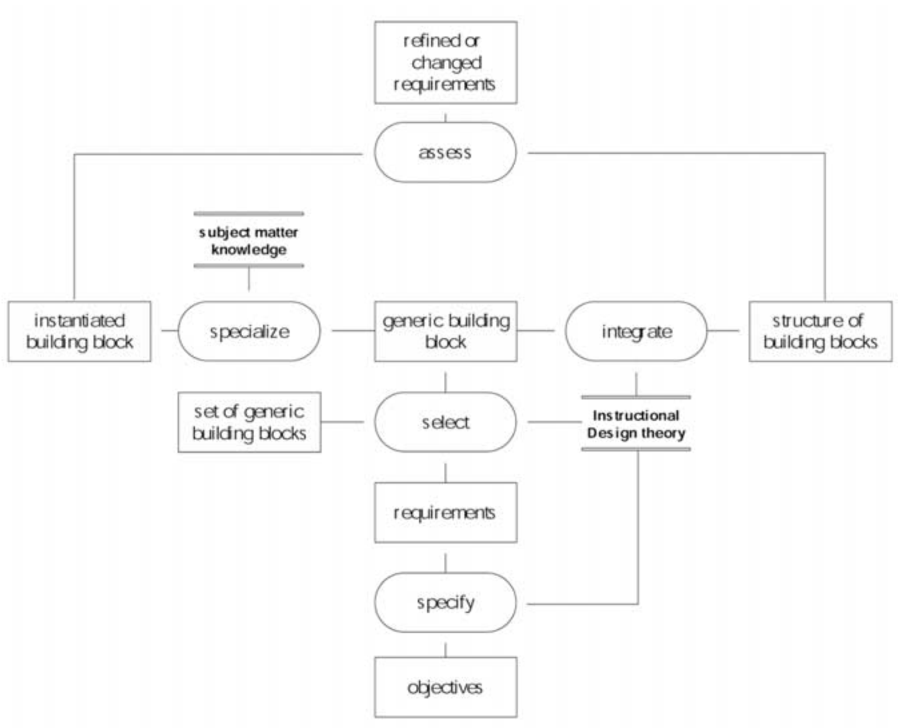

refined or changed requirements
assess
subject matter knowledge
instantiated building block
specialize
generic building block
integrate
structure of building blocks
set of generic building blocks
select
Instructional Design theory
requirements
specify
objectives

Figure 3. A generic model of authoring based on building blocks; components corresponding to elements present in SIMQUEST are indicated in grey.

knowledge (support knowledge) needed to carry out the associated reasoning steps. As the focus in this section is on the reasoning steps specify, select, specialize, integrate, assess, they are elaborated below.

Specify. On the basis of the design objectives a set of requirements is specified. Design objectives are formulated in terms of a state that needs to be achieved. In our case the author wants the learner to understand certain subject matter knowledge. This is usually formulated in terms of instructional objectives, the knowledge and skills that should be acquired by the learner. In the specification process the objectives are translated into requirements, which describe the behavior of the system and how this behavior should lead to the accomplishment of the objectives.

Select. In configuration design, predefined solutions – building blocks – are available for making a system behave in certain way. An important step in the design process is selecting the appropriate solution for a particular sub-problem; i.e. requirement. The selection step is constrained by the available building blocks; this requires knowledge about the different types of building blocks. The selection step is supported by knowledge of instructional design. The selection step delivers a generic building block – a template – which needs to be filled with subject matter knowledge, and which needs to be combined with other building blocks to achieve overall system behavior.

Specialize. Filling a selected generic building block with subject matter knowledge is called specialization of a building block. As indicated, subject matter knowledge plays a support role in this step.

Integrate. In the integrate step, the building block from the specialize step is related to other building blocks. The overall structure of building blocks makes up the application. Again this process is supported by knowledge about instructional design.

Assess. Both the specialization and integration steps produce solutions for the design problem, both at a different level. Whether the solutions fulfill the initial requirements needs to be assessed. The assessment step often leads to the alteration or refinement of requirements, because chosen solutions bring new insights to the surface.

The non-linear, cyclic nature of the reasoning process described above is not directly represented in Figure 3, as, according to CommonKADS, the control over the reasoning process is written out in a separate layer. For Figure 3 this would entail a repeat ... until like structure in which the reasoning steps and their inputs and outputs appear in a kind of pseudo code.

The generic model in Figure 3 reasons of course about a domain, which provides the 'content' to the elongated ovals and rectangles. The authoring products constitute the *domain* of the process model described in Figure 3. The next section is devoted to the modeling of this domain, i.e. the authoring products.

Modeling the authoring products

The entities bridging the gap between the authoring domain and the authoring environment are the (intermediate) authoring products, labeled 'knowledge models' in Figure 2.

In order to develop these models, the domain, or the 'design space', has to be carved up in smaller chunks. This makes control over the authoring process and products easier. There are many different ways to do this. The authors simply adopt the four models distinguished by Kuyper (1998) and refer the interested reader to this source for an in depth discussion of the arguments for choosing these four. Below a short description each of the models will be given.

Instructional model

The instructional model covers those aspects that are relevant from the learning point of view. It represents the most important dimensions mentioned in the literature:

1. *The structure of concepts*: the decomposition of learning goals into subgoals
2. *Partial solutions*: learning goals and instructional actions can be seen as design objects having generic attributes which must be instantiated for a specific context
3. *Concept dependencies*: the relation between the instruction and the simulation's complexity
4. *Time relations*: the phasing and timing of instructional actions

Simulation model

The simulation model is directed at developing the simulation but takes into account that this model should be used for instruction. It offers guidance to represent different relevant aspects of a simulation model for instruction. Firstly, aggregates of variables support a structured decomposition of the model, that enables the representation of concepts that can be used in an instructional context for reasoning about system behavior. Secondly, critical states identify important shifts in system behavior. Thirdly, making explicit the modeling assumptions indicates relevant constraints of the application of the formal model. All three aspects serve as important resources for the decomposition of learning goals and inspire the design of experiments.

Interaction model

Another aspect of authoring is the visualization and manipulation of the simulation and instructional support. Both the simulation model and the instructional model offer little information about the knowledge required for authoring the interaction between their elements and the student. To bridge the gap with the applied knowledge used in authoring with a specific authoring environment, it is necessary to align to the interaction possibilities this environment provides. The interaction model captures the interaction elements that are available to the learner and models how through the use of scenarios a preliminary understanding of application functionality can be developed.

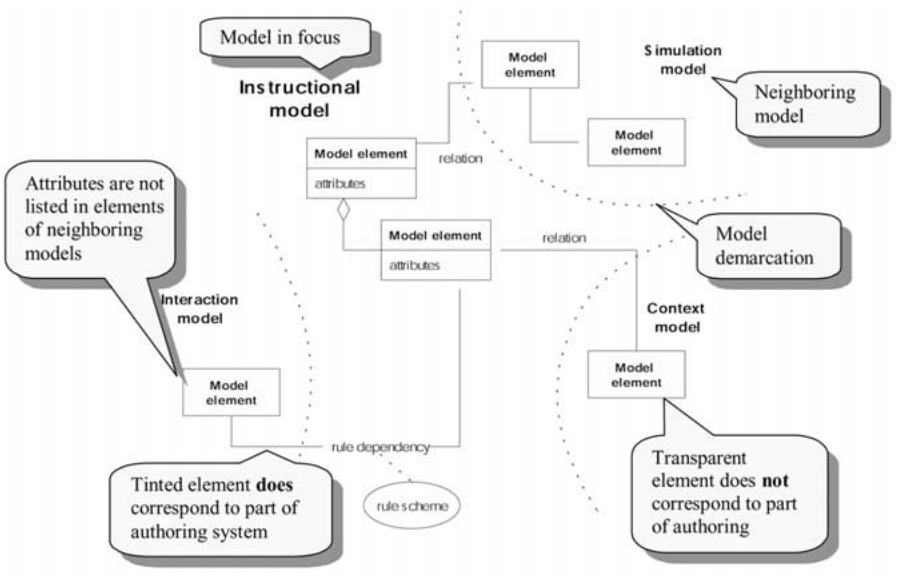


Figure 4. General frame for representing model structures.

Context model

Although we do not aim to model the organizational context in which an instructional simulation is applied, we cannot neglect the context in which it is developed. Educational programs are formally set up according to attain objectives specified by national institutes that make educational policies. To integrate instructional means such as an instructional simulation into an educational program it has to comply with one or more of the objectives that are applicable to that program.

The description of the models given above is discursive in nature; it's hard to figure out what their structure is. As any model is a set of concepts and the relations between these concepts, making the models more precise entails specifying the concepts and their relations. In the framework of SIMQUEST the four models sketched above have been modeled in great detail (see for comparable models in CommonKADS, De Hoog, 1997) to permit a consistent mapping of the intermediate products on the behavior and appearance of the authoring support wizard. In order to illustrate this preciseness, one model will be shown in more detail. For the other models the reader is referred to Kuyper (1998). The graphical presentation of models will be depicted as shown in Figure 4.

In Figure 4 rectangles represent the concepts in a model, a concept can have attributes, which describe relevant features of the concept. Relations between concepts are of the following types:

a) *Part-of-relation* (a diamond on the line relating concepts).
b) *Sub-type relation* (an open arrowhead on the line relating concepts, not present in the examples).
c) *Directional binary relation* (a small black arrowhead on the relation plus the name of the relation).
d) *Rule schema relation* (two small black arrowheads, interrupted by text and linked to an oval).

This last relation type are schemas that are used to provide knowledge when the nature of an element will strongly influence the nature of another. It can be seen as an annotation on a relation between two concepts.

Although these models demarcate separate parts in the domain there are many dependencies between them. To indicate the dependencies between models in the graphical notation, the relevant model elements of neighboring models will be displayed as well. The demarcations between the models are indicated by a dotted line. Also the model elements that correspond to building blocks in the authoring environment are tinted in grey, while model elements that are part of the authoring knowledge but do not reside in the support environment are transparent. We will take one of the simpler models, the context model, as an example. This context model describes the relation between the application and the environment in which it has to function.

The organizational context in which we considered instructional simulation development to take place is Senior Secondary Vocational Education. In this type of education, classes mainly consist of formal lecturing and practicals. Education is organized according to educational objectives that are nationally determined. Within the scope of these instructional objectives, teachers determine what learning material is used and how to test the accomplishment of specific instructional objectives. Instructional simulations can be used in this context to bridge the gap between theory and practicals. Subject matter theory is often very abstract for learners and lacks a dynamical or visual representation. Practicals help to train procedures and learning about real-life contexts, but often fail to connect to theoretical insights. For example, in the domain of construction mechanics the understanding of stressing a beam requires a complex visualization of stress events present in the beam. The theory can deliver formulas to calculate certain measures needed in practice and in practicals measurement instruments can be used to show the value of these measures, but still the underlying model of what these measures represent and how these are related might be unclear for the learner. Instructional simulations can deliver just these insights by an abstract

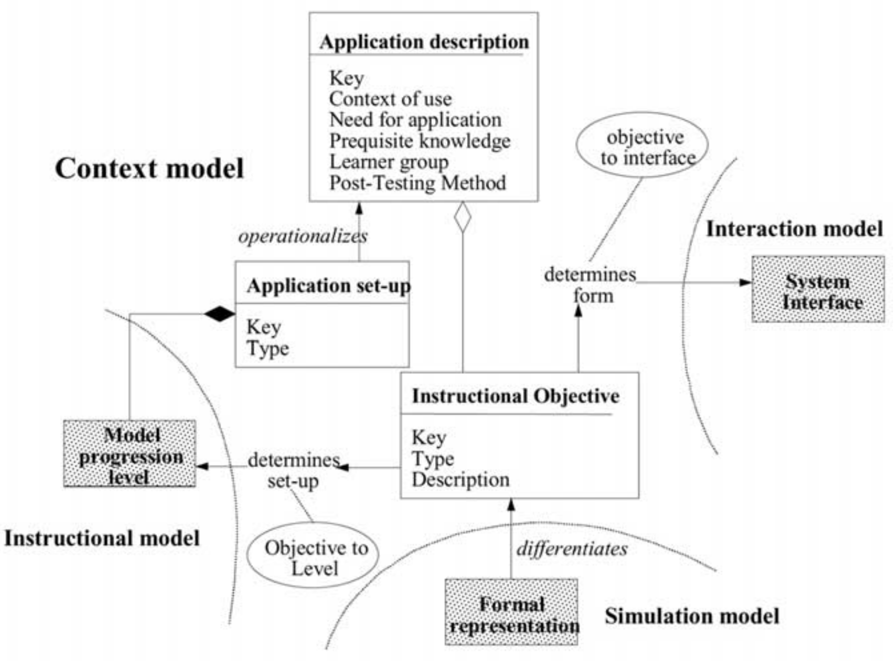


Figure 5. Structure of the context model.

visualization of reality and as such are useful tools in education. A simulation can be used to show the stress in the beam by means of changing colors and by showing measured values at the right places. The introduction of instructional simulations in the curriculum is part of a trend towards ICT-based (Information and Communication Technology) learning and fits in with views on education which advocate independent learning.

Instructional simulations are not meant to replace formal lecturing, but rather to enhance certain parts of it with dynamic representations of hard to show underlying processes. This has repercussions for support teaching and instruction. Support needs to enable the integration of an instructional simulation with the current practice of teaching. An instructional simulation development environment has to support the use of similar terminology, formulas and visual metaphors to fit the choice of existing textbooks. In addition it has to provide for visualization building blocks that transfer to the situation used in practicals. It has to fit in with the instructional objectives imposed on students. Therefore, support should enable the mapping from these objectives to learning goals. The rationale behind this model is therefore that context requirements need to be related to the general characteristics of the application.

The context model is depicted in Figure 5. The context model consists of the following concepts: application description, instructional objective and application set-up. Each of these concepts is characterized by a set of attributes (e.g. 'Key' in application description), which can be seen as a more detailed description of the concept. Furthermore, in the context model the concepts are connected to each other through relations like 'operationalizes'.

Concepts

1. The *Application description* is an aspect that is used in the early phases of development for communication between different parties in development; i.e. subject matter experts, instructional designer, project managers, etc. It states the overall objectives of the application and the need that it satisfies. It sketches the target learner group which is going to be served by the application and describes the prerequisite knowledge to enter the application and how it is embedded in current learning material. For Middle Vocational Training the 'need for application' attribute could be filled by something that was written above: Instructional simulations can be used in this context to bridge the gap between theory and practicals. Subject matter theory is often abstract for learners and lacks the support of a dynamic or visual representation. Practicals help to train procedures and learning about real-life contexts, but often fail to connect to theoretical knowledge.
2. The *Instructional objective* represents the overall goal that needs to be reached by the learner. Instructional objectives are determined nationwide for particular educational programs and have a fixed structure.
3. The *Application set-up* describes the contents of the application as it is presented to the student and presents an overall framework for building the application.

Relations

There are two types of relations in Figure 5: unsupported relations and rule schema supported relations. The unsupported relations are:
1. Application set-up *operationalizes* Application description. The application description is a mission statement about the application; it does not contain any design decisions. The application set-up does embody design decisions based on the instructional objectives in the Application description.
2. Formal representation *differentiates* Instructional objectives. When the details of the formal representation are investigated different topics of interest arise which specify the original objective in greater detail.

Two-rule schema supported relations in Figure 5 are:

1. Instructional objective *determines form* of system interface. An associated support rule is: if the procedure to be learned is physical then you should choose an enactive (3 dimensional) or iconic (video or graphics) representation form (cf. Reigeluth & Schwarz, 1989)
2. Instructional objective *determines set up* of model progression level. An associated support rule is: if the objective is to learn a principle then you should require the learner to manipulate examples, observe cause and effects, and figure out the principles during the acquisition stage and play a role applying the principles during the application stage (cf. Reigeluth & Schwarz, 1989).

The context model sketched above is the simplest model that was developed; the other model structures are far more complex. The instructional model, for example, contains 16 concepts and 25 attributes. Relations in the model and between other models number more than twenty. The important thing to realize is that this level of detail and precision in the models permits a far clearer understanding of the nature of the authoring products than the discursive description given earlier. It is this precision that permits a theoretically underpinned design of the authoring wizard, which will be the topic of the next section.

Mapping models on support: The SimQuest Wizard

In the previous two sections models were presented for the authoring process and the authoring products that form the bridge between the authoring domain and the authoring tool, as shown in Figure 2. This leaves the rightmost part for further elaboration: mapping the intermediate products and the generic process on the SimQuest environment. To support this a 'Wizard' was developed. The Wizard addresses three parts: *mapping the products*, *mapping the process* and *connecting products and process*.

Mapping the products

The models have three main features (see Figure 4), which should be taken into account when mapping them on a support environment:

a) Relations, in particular part-of or hierarchical relations between concepts
b) Attributes of concepts
c) Rule schemes

Below, their implementation in the SimQuest Wizard is elaborated.

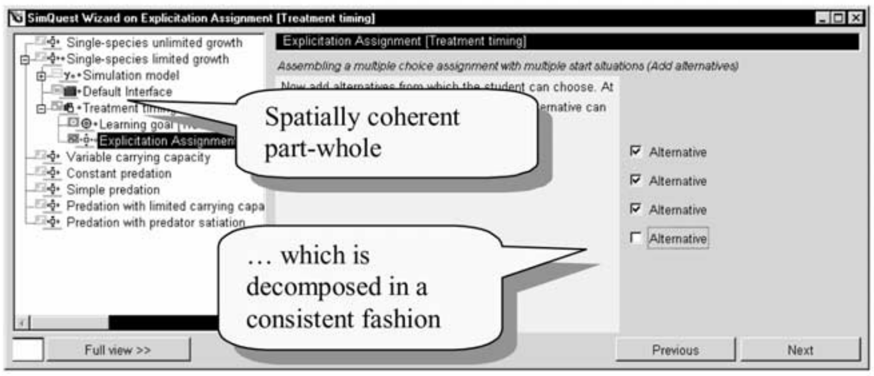

SimQuest Wizard on Explicitation Assignment [Treatment timing]
Single-species unlimited growth
Single-species limited growth
Simulation model
Default Interface
Treatment timing
Learning goal
Explicitation Assignment
Variable carrying capacity
Constant predation
Simple predation
Predation with limited carrying capa
Predation with predator satiation
Explicitation Assignment [Treatment timing]
Assembling a multiple choice assignment with multiple start situations (Add alternatives)
Now add alternatives from which the student can choose. At
ernative can
Alternative
Alternative
Alternative
Alternative
Full view >>
Previous
Next
Spatially coherent part-whole
… which is decomposed in a consistent fashion

Figure 6. Representation of a part-whole structure in the wizard environment.

1. *Hierarchy, part-of-relations*

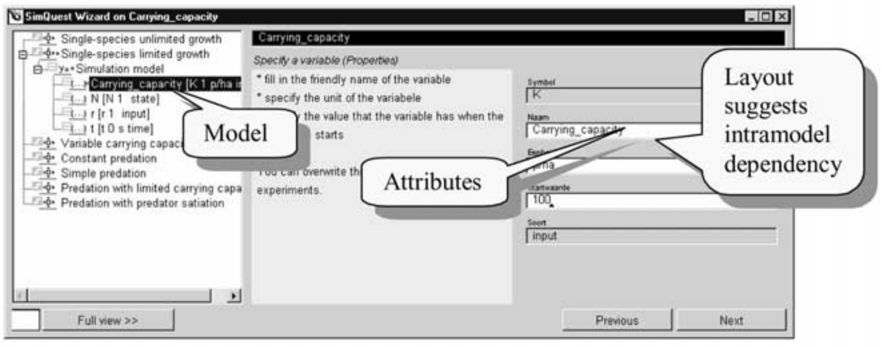
An important feature of knowledge/product models is that they incorporate the hierarchical structure of a product. According to Norman's model (Norman, 1986) this represents a decomposition of working goals. A way to afford this decomposition of goals is by representing the conceptual part-whole structure described by product models as an on-screen part-whole structure in the system interface. In the SIMQUEST wizard environment a specific textual tree structure is used, that was available in the VisualWorks® programming environment[2] (see Figures 1 and 6).

The part-whole structures in product models represent coherent groups of concepts based on studies of best practices. The coherency in product models should be reflected in the organization of application components; i.e. a coherent set of components should also be presented in a spatially coherent way. Figure 6 shows how the part-whole structure is embedded in the SIMQUEST wizard environment.

The product models described in the Section Modeling the Authoring Products represent intermediate as well as final authoring products. If the author conceptually works on these products they should be explicitly represented in the system interface. A good example is a learning goal; although a learning goal is not necessary to run the system, every author at least has given some thought to the instructional purpose of assignments and experiments.

2. *Model attributes and intramodel dependencies*

The attributes of a model concept refer to its generic properties that reoccur in different instructional situations. The detailed models that have been developed, like the Context model in Figure 5, predefine these generic properties. Each attribute of a concept in a model has to be presented explicitly in the


SimQuest Wizard on Carrying_capacity
Model
Attributes
Layout suggests intramodel dependency
Full view >>
Previous
Next

Single-species unlimited growth
Single-species limited growth
Simulation model
Carrying_capacity [K 1 p/ha i
N [N 1 state]
r [r 1 input]
t [t 0 s time]
Variable carrying capaci
Constant predation
Simple predation
Predation with limited carrying capa
Predation with predator satiation

Carrying_capacity

Specify a variable (Properties)

* fill in the friendly name of the variable
* specify the unit of the variabele
the value that the variable has when the starts

You can overwrite th experiments.

Symbol: K
Naam: Carrying_capacity
p/ha
100
Soort: input

Figure 7. The representation of model concept attributes in the wizard environment.

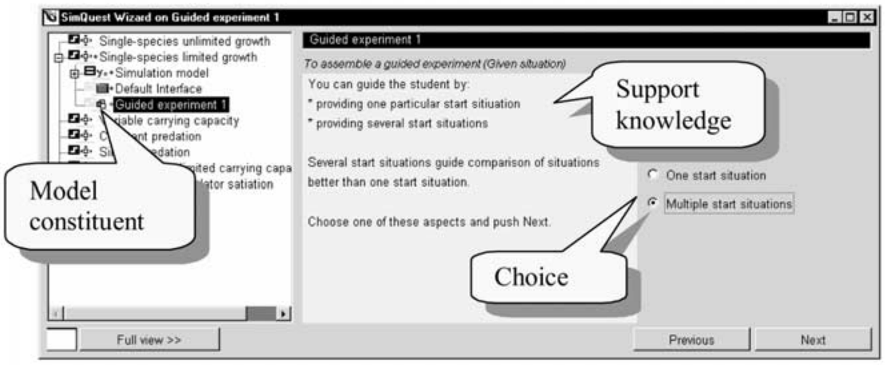
system interface. In the wizard environment, the attributes of an application component are represented by text fields (see Figure 7).

Furthermore it is of importance that the role of the attribute can be related to the context in which it is used. In the wizard environment text fields are presented as a result of the selection of a component in the application structure (shown in the left-hand side of Figure 7).

The intramodel (i.e. within one model) dependencies are dependencies between attributes that are part of the same model concept. These dependencies express prerequisite or suggestive relations; information to specify one attribute is required or desired to specify the information in another attribute; i.e. thinking about the unit of a variable helps estimating the initial value. The dependencies are represented in the SIMQUEST wizard environment by the order and layout of the fill-in fields. The lay out of the attributes (rightmost part) in Figure 7, from top to bottom, indicate that before the author can specify the Start value of a variable (Carrying capacity), the Unit ('Eenheid') for the variable has to be specified. These dependencies are either strong or weak: when they are strong a fill-in field is not accessible before the predecessor field is entered.

3. *Rule schemes*

Rule schemes represent design dependencies, which are *inter*model dependencies (i.e. between models). An example of such a dependency in the authoring process might be: in an introductory situation offer learners a set of start situations so that they do not have to think of these settings themselves. In an advanced situation only give dedicated start situations. In the context of SIMQUEST this dependency holds, for example, between the role of an experiment and the type of assignment that is chosen. Providing the author with the relevant knowledge to apply such a rule is a form of decision support.



Guided experiment 1

To assemble a guided experiment (Given situation)

You can guide the student by:
* providing one particular start sitiuation
* providing several start situations

Several start situations guide comparison of situations better than one start situation.

Choose one of these aspects and push Next.

( ) One start situation
(•) Multiple start situations

Figure 8. The representation of rule schemes in the wizard environment.

In addition to the mapping rules that hold for representing decision rules, it can be said that knowledge modeling can be used to make implicit expert knowledge about design rules explicit and helps to transfer part of the design task from the user to the system.

The author has to be made aware that there is a relevant decision to make, therefore it has to be explicitly represented. In the SIMQUEST wizard this is achieved by means of a separate step in the wizard sequence. The decision support should be presented at the time and in the context where the decision has to be made.

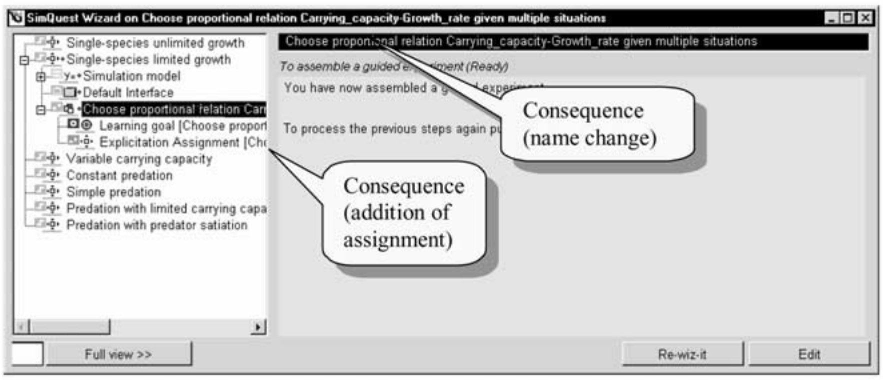
In Figure 8 the effect of a rule schema being operational in the Wizard is shown.

In Figure 8 the author has decided to create a guided experiment. In the middle part of the window the support knowledge available for this step appears. The right part of the window gives the decision options (one or multiple start situations) which are presented in the middle part.

After the decision has been taken, the effect should be made explicit to the author. In the situation depicted in Figure 9 a certain type of assignment (Explicitation assignment) is suggested, based on the decision in Figure 8 to have a Guided experiment with multiple start situations.

Mapping the authoring process onto the wizard

As was described in Sections 1 and 2, the main steps in working with components supported by SIMQUEST are selecting a concept, instantiating its attributes and relations, and linking it to other concepts. Especially the selection and linking steps are usually hard to support because they are dependent on a specific application context. The use of predefined model



SimQuest Wizard on Choose proportional relation Carrying_capacity-Growth_rate given multiple situations

- Single-species unlimited growth
- Single-species limited growth
  - Simulation model
  - Default Interface
  - Choose proportional relation Carr
    - Learning goal [Choose proport
    - Explicitation Assignment [Cho
- Variable carrying capacity
- Constant predation
- Simple predation
- Predation with limited carrying capa
- Predation with predator satiation

Choose proporti…nal relation Carrying_capacity-Growth_rate given multiple situations

To assemble a guided e… …ment (Ready)

You have now assembled a g… …d experiment

To process the previous steps again pu

Consequence (name change)
Consequence (addition of assignment)
Full view >>
Re-wiz-it
Edit

Figure 9. The representation of consequences of applying a rule in the wizard environment.

structures makes a way of working possible; i.e., the expansion of models and the specialization of its concepts and relations.

In a model a part-whole structure is decomposed until a model concept can be described solely by its attributes. Creating a concept's parts is labeled as an *expansion* activity, while instantiating a concept's attributes is called *specialization.* In the SIMQUEST wizard environment expansion results in the addition of components in the application structure, while specialization is supported by a plan of steps in which fill-in fields can be filled (see left-hand part and right-hand part of Figure 7 respectively). When to offer expansion and when to offer specialization support can be deduced from the decomposition of the part-whole structure. The different manipulation activities should be consistently represented and visibly indicated in the system interface. The expansion activity in the wizard environment is implemented by providing a checkbox with a tick mark and pushing the button *Next* or *Finish*. The specialization activity is implemented by filling text fields and pushing the button *Next* or *Finish.*

The wizard environment operationalizes this way of working, but it requires an adaptation of the original process model as represented in Figure 3. This led to a new process model shown in Figure 10.

This process model is informative for the design of the wizard environment. How some of the reasoning steps from Figure 10 are supported in the wizard environment is shown in Figure 11 (the expand step) and Figure 12 (the specialize step).

Connecting product and process: model states

In the modeling approach of CommonKADS the notion of model states is used to connect a product view (models) with a process (authoring) view.

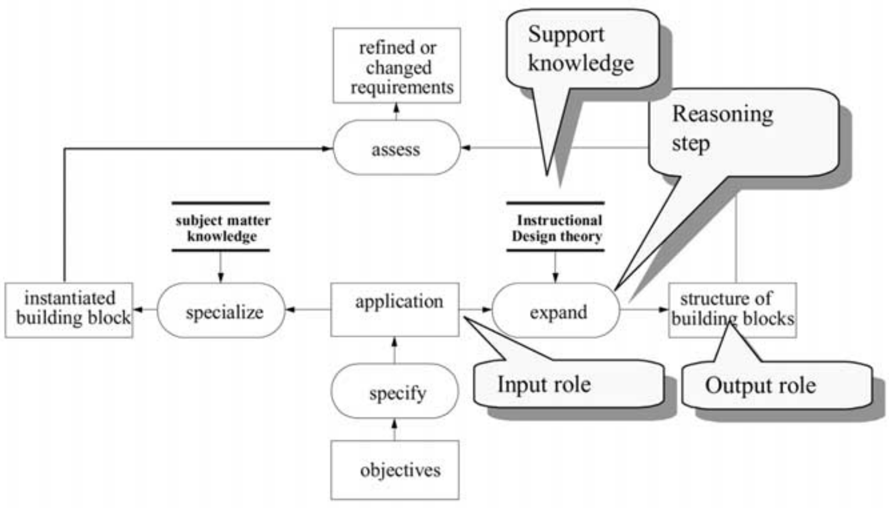
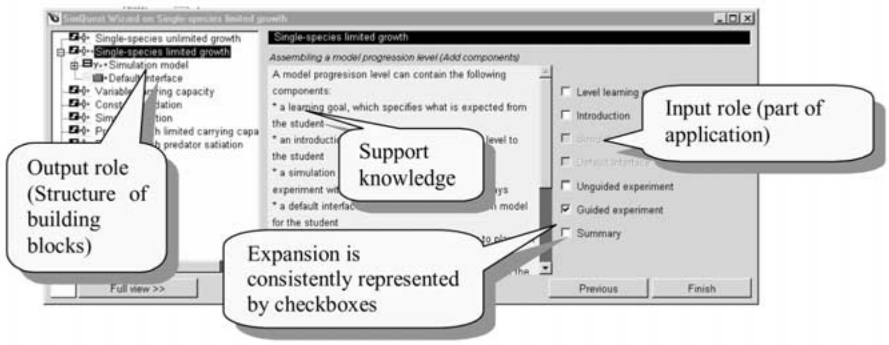


Figure 10. A revised authoring process model operationalized by the wizard environment.


SimQuest Wizard on Single-species limited growth
Full view >>
Previous
Finish
Output role (Structure of building blocks)
Support knowledge
Input role (part of application)
Expansion is consistently represented by checkboxes

Single-species unlimited growth
Single-species limited growth
Simulation model
Default interface
Variable carrying capacity
Const dation
Sim tion
P h limited carrying capa
h predator satiation

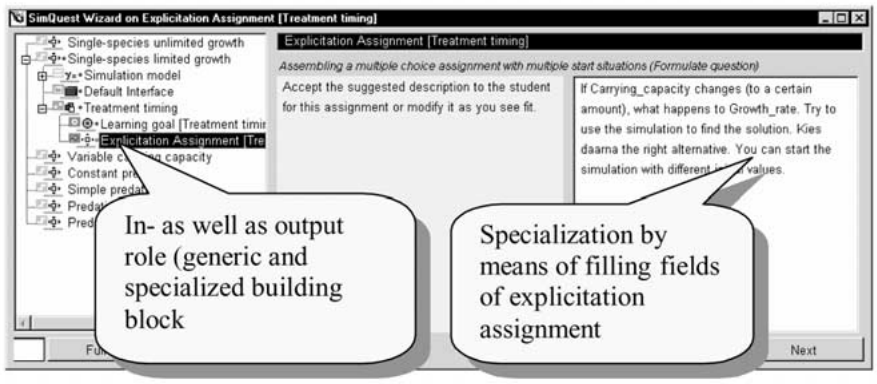
Single-species limited growth

Assembling a model progression level (Add components)

A model progresison level can contain the following components:
* a learning goal, which specifies what is expected from the student
* an introducti level to the student
* a simulation
experiment wi ys
* a default interfac n model for the student

☐ Level learning
☐ Introduction
☐ Simu
☐ Default interface
☐ Unguided experiment
☑ Guided experiment
☐ Summary

Figure 11. The representation of the expand step in the wizard environment.


Full
Next
In- as well as output role (generic and specialized building block
Specialization by means of filling fields of explicitation assignment

SimQuest Wizard on Explicitation Assignment [Treatment timing]

Single-species unlimited growth
Single-species limited growth
Simulation model
Default Interface
Treatment timing
Learning goal [Treatment timir
Explicitation Assignment [Tre
Variable c g capacity
Constant pr
Simple preda
Preda
Pred

Explicitation Assignment [Treatment timing]

Assembling a multiple choice assignment with multiple start situations (Formulate question)

Accept the suggested description to the student for this assignment or modify it as you see fit.

If Carrying_capacity changes (to a certain amount), what happens to Growth_rate. Try to use the simulation to find the solution. Kies daarna the right alternative. You can start the simulation with different values.

Figure 12. The representation of the specialize step in the wizard environment.

A model state should be seen as a kind of snapshot in time reflecting the condition of a model and its components (concepts, attributes and relations). From a particular model state only a limited number of steps is possible, thereby affording and constraining process execution. Executing a step leads to changes in the model state (and possibly others), which means a refinement of the product. This new model state opens up new steps, which will lead to a further product refinement. The process therefore generates its own progress possibilities. This approach is especially suited for providing guidance for ill-structured tasks in which the precise execution of activities is difficult to prescribe. By setting out the goal structure of the task and by providing a flexible computational architecture for refining it, activities can be executed as new insights occur. This provides for flexibility[4] of the authoring process, as proposed by De Hoog et al. (1994).

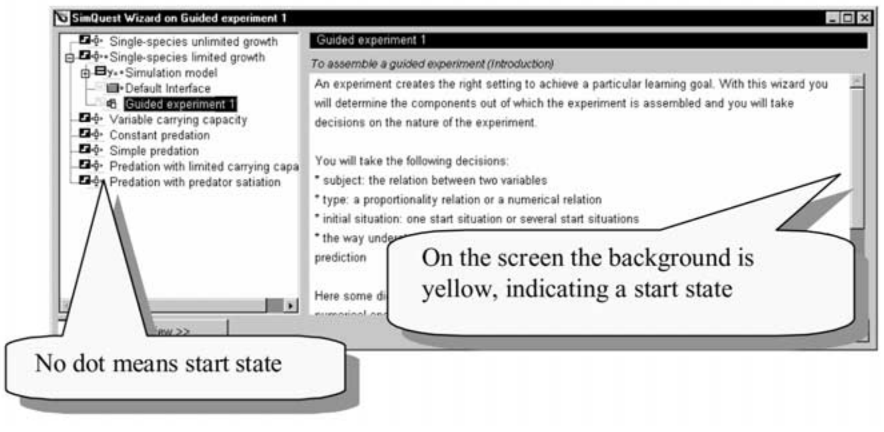
The expansion of an application component in the wizard environment, is indicated by a plus/minus sign in front of its label. The sign indicates that its subcomponents are displayed (minus) or hidden (plus). If the components do not have any subcomponents, and are not expanded, no sign is visible. This provides for a high level overview of the state of expansion of a model.

The Wizard plan for each model component, which entails expansion as well as specialization, can be in one of three states, *start*, *transition*, and *ready*. In the *start* state the specialization/expansion of a component has not begun yet. In the *ready* state all steps have been taken. Any step in between renders the process in a *transition* state. These states are indicated by the different colors of the wizard pages (steps), *yellow* for *start*, *green* for *transition* and *red* for *ready*.[5] In addition each component in the application structure has a colored dot in front of its label, corresponding to its state. The latter aspect takes care that the state of the component is recognizable, even when it is not selected.

Figure 13, Figure 14 and Figure 15 show how this notion of 'colored' states is implemented in the Wizard environment. The task is to create a guided experiment.

In Figure 13 the start of creating a Guided experiment is shown. The right-hand part of the window gives an overview of the activities and decisions the author has to make. This prepares the author for what the authoring environment is expecting as input.

In Figure 14 the author has started the work on the building of a Guided experiment. The first attributes are instantiated (right-most part of the window). The Green dot in the left-most part indicates that this concept is now under construction. As interrupts occur quite frequently during authoring, for example because the author realizes that some other work has to be done first or simply the working hours are past, the Green dot is a reminder of work to


SimQuest Wizard on Guided experiment 1
Single-species unlimited growth
Single-species limited growth
Simulation model
Default Interface
Guided experiment 1
Variable carrying capacity
Constant predation
Simple predation
Predation with limited carrying capa
Predation with predator satiation
On the screen the background is yellow, indicating a start state
No dot means start state

Guided experiment 1

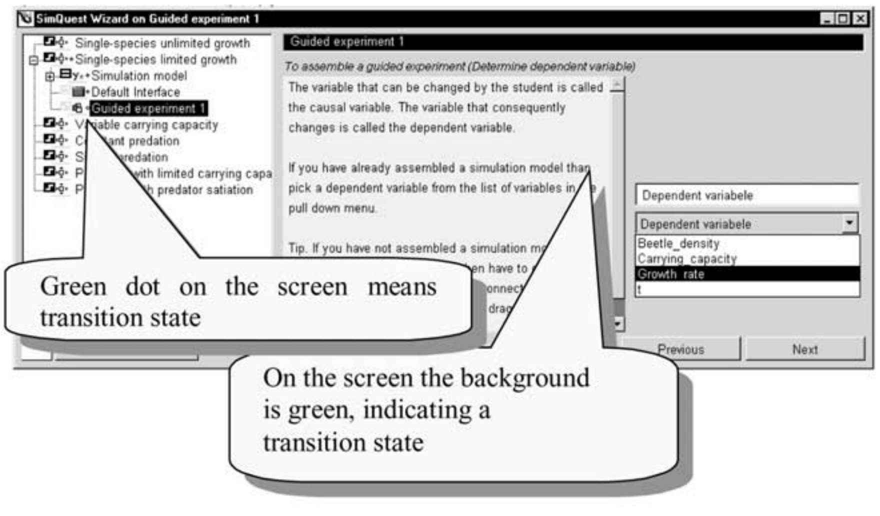
To assemble a guided experiment (Introduction)

An experiment creates the right setting to achieve a particular learning goal. With this wizard you will determine the components out of which the experiment is assembled and you will take decisions on the nature of the experiment.

You will take the following decisions:
* subject: the relation between two variables
* type: a proportionality relation or a numerical relation
* initial situation: one start situation or several start situations
* the way unde
prediction

Here some di

Figure 13. Start state of guided experiment.


SimQuest Wizard on Guided experiment 1
Single-species unlimited growth
Single-species limited growth
Simulation model
Default Interface
Guided experiment 1
Dependent variabele
Beetle_density
Carrying_capacity
Growth_rate
t
Previous
Next
Green dot on the screen means transition state
On the screen the background is green, indicating a transition state

Guided experiment 1

To assemble a guided experiment (Determine dependent variable)

The variable that can be changed by the student is called the causal variable. The variable that consequently changes is called the dependent variable.

If you have already assembled a simulation model than pick a dependent variable from the list of variables in the pull down menu.

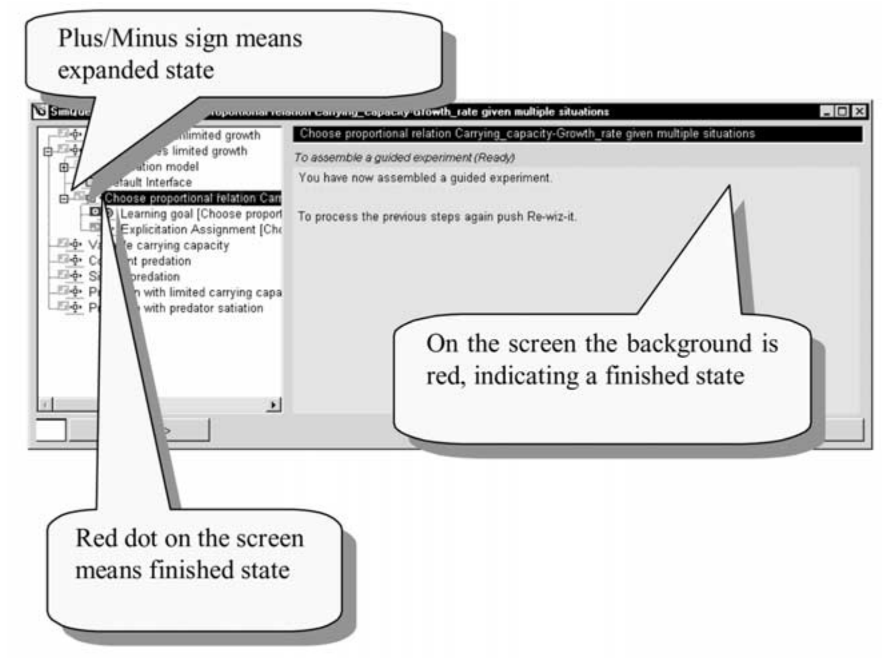
Tip. If you have not assembled a simulation m

Figure 14. Transition state of guided experiment.

do. This is important when applications are growing and it becomes hard to keep track of what has been done and still has to be finished.

In Figure 15 the work on the Guided experiment is finished: a suitable assignment (explicitation) has been constructed and the Red dot in the left-hand part of the window shows that the concept has been fully instantiated. In this way the author can always see at a glance what the state of the work is.

A weak part of this representation is that the transition state is not differentiated according to the type of activity; i.e. specialization or expansion. Nor


Plus/Minus sign means expanded state
On the screen the background is red, indicating a finished state
Red dot on the screen means finished state
nlimited growth
s limited growth
ation model
efault Interface
Choose proportional relation Car
Learning goal [Choose proport
Explicitation Assignment [Ch
e carrying capacity
nt predation
predation
n with limited carrying capa
with predator satiation

Choose proportional relation Carrying_capacity-Growth_rate given multiple situations

To assemble a guided experiment (Ready)

You have now assembled a guided experiment.

To process the previous steps again push Re-wiz-it.

Figure 15. Finished state of guided experiment.

does the state of an aggregate component reveal any information about the states of its subcomponents; i.e. in principle an aggregate component is *ready* only when all of its subcomponents are *ready*. This must be seen as an issue for further development.

Summary and conclusions

The approach sketched in this paper is an example of how the support for authoring for simulation based educational software can be grounded in a specific, semi-formal theory. The ‘look and feel’ of this kind of support should not be derived from incidental, not to say accidental, hunches of interface designers, but has to be based on an in depth analysis of the *what* and *how* of authoring. This analysis can be approached fruitfully as a kind of knowledge engineering activity and can benefit from advances made in this area by the CommonKADS methodology. By taking up the basic ideas from this methodology and tailoring it to the specific area under consideration, a coherent set of intermediate products (models) and a generic process model could be built. The level of detail achieved permits consistent mapping on a Wizard which implements the basic processes and structures resulting from the analysis. This Wizard facilitates the complex authoring task by specifying products and actions to be build, and provides tailored advice for this task. As a consequence the behavior and lay out of the Wizard is derived from a theory instead of non-specific interface design guidelines which abound in the literature (see e.g., Nielsen & Mack, 1994). This is not to say that these guidelines have no value, but they should be seen as an *adjunct* to an in-depth analysis as presented in this paper. This analysis can also be seen as a case of modeling for *usability* and *interaction design* before one can move to *interface design*.

The product and process models that were developed (though not all of them were described in this paper) prove that authoring design products and processes are amenable to more 'formal' definitions than the discursive descriptions one usually encounters in the literature. The need to define fairly general model structures for important areas of concern during design, forces one to think hard about the precise nature of the concepts and relations between concepts that constitute these models. In this respect an important characteristic of our approach as compared to more traditional approaches (e.g., Romiszowski, 1981) is that the results of the analyses directly feed into the instructional material developed. Another difference with traditional approaches is our emphasis on products instead of processes (see e.g., Smith & Ragan, 1999). What our approach does not give is advice on how to take specific instructional decisions (as detailed decisions within the overall SIMQUEST philosophy). A separate module within SIMQUEST (the advice module, see Limbach et al., 1999) houses the accompanying instructional design theory (see Reigeluth, 1999) for discovery learning.

We do not intend to say that the models we have used are to be seen as the final word. We can and will not claim these this are 'the' models for now and forever, because they are at least to some extent specific for the SIMQUEST context. However, we claim that the modeling approach followed can be and should be re-used whenever design and development of authoring systems for educational software is on the agenda.

Notes

1. A comparable approach for teaching has been advocated by Van Marcke (1998) in GTE (Generic Tutoring Environment).
2. www.objectshare.com.
3. The topic of the next series of Figures is population ecology (predator-prey models).
4. A product-oriented way of working does not necessarily mean that alternative orders of activities are always possible. The structures of products may constrain each other in such a way that only one order may apply.
5. Cf. traffic lights where Green indicates Proceed and Red indicates Stop.

References

Breuker, J.A. & Van de Velde, W.A. (1994). *The CommonKADS Library for Expertise Modelling*. Amsterdam: IOS Press.

De Hoog, R., De Jong, T. & De Vries, F. (1994). Constraint driven software design: An escape from the waterfall model. *Performance Improvement Quarterly* 7: 48–63.

De Hoog, R. (1997). CommonKADS: knowledge acquisition and design support methodology for structuring the KBS integration process. In J. Liebowitz & L. Wilcox, eds, *Knowledge Management and Its Integrative Elements*, pp. 129–142. Boca Raton: CRC Press.

De Jong, T., Härtel, H., Swaak, J. & Van Joolingen, W. (1996). Support for simulation-based learning; the effects of assignments in learning about transmission lines. In A. Díaz de Ilarazza Sánchez and I. Fernández de Castro, eds, *Computer Aided Learning and Instruction in Science and Engineering*, pp. 9–27. Berlin: Springer Verlag.

De Jong, T., Van Joolingen, W.R., Swaak, J., Veermans, K., Limbach, R., King, S. & Gureghian, D. (1998). Self-directed learning in simulation-based discovery environments. *Journal of Computer Assisted Learning* 14: 235–246.

De Jong, T., Martin, E., Zamarro, J-M., Esquembre, F., Swaak, J. & Van Joolingen, W.R. (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.

Kuyper, M., De Hoog, R., Van der Hulst, A., Van Doorn, F. & Pieters, J. (1995). *Final Report on Pilot Testing of the Authoring Toolkit*. Technical Report, Dept. of Social Science Informatics, University of Amsterdam, Deliverable D33 of DELTA project SMISLE D2007.

Kuyper, M. (1998). *Knowledge Engineering for Usability. Model-mediated Interaction Design of Authoring Instructional Simulations*. Ph D. Thesis, University of Amsterdam.

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.

Nielsen, J. & Mack, R., eds (1994). *Usability Inspection Methods*. New York: Wiley.

Norman, D.A. (1986). Cognitive engineering. In D.A. Norman & S. Draper, eds, *User Centered System Design: New Perspectives on Human-Computer Interaction*, pp. 31–63. Hillsdale, NJ: Lawrence Erlbaum Associates.

Reigeluth, C.M. & Schwartz, E. (1989). An instructional theory for the design of computer based simulations. *Journal of Computer-based instruction* 16: 1–10.

Reigeluth, C. (1999). *Instructional-design Theories and Models*. Mahwah, NJ: Erlbaum.

Romiszowski, A.J. (1981). *Designing Instructional Systems*. London: Kogan Page.

Smith, P.L. & Ragan, T.J. (1999). *Instructional Design*. New York: John Wiley.

Schreiber, A.Th., Akkermans, J.M., Anjewierden, A.A., De Hoog, R., Shadbolt, N.R., Van de Velde, W. & Wielinga, B.J. (2000). *Knowledge Engineering and Management. The CommonKADS Methodology*. MIT Press.

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.

Top, J.L. & Akkermans, J.M. (1994). Engineering modelling. In J.A. Breuker & W. van de Velde, eds, *The CommonKADS Library for Expertise Modelling*, pp. 265–304. Amsterdam: IOS Press.

Van Joolingen, W.R., King, S. & De Jong, T. (1997). The SIMQUEST authoring system for simulation-base discovery environments. In B. du Boulay & R. Mizoguchi, eds, *Knowledge and Media in Learning Systems*, pp. 79–87. Amsterdam: IOS.

Van Marcke, K. (1998). GTE: An epistomological approach to instructional modeling. *Instructional Science* 26: 147–191.