The SimQuest Authoring System for Simulation-Based Discovery Learning

Wouter van Joolingen1, Simon King2 and Ton de Jong1
1University of Twente, P.O. box 217 7500 AE Enschede the Netherlands
email: [joolingen, jong1]@edc.utwente.nl
2EDS Ingevision, 4 avenue Pablo Picasso 92024 Nanterre Cedex France
email: king@framenetec.fr

The current paper discusses SimQuest, an authoring system for simulation-based discovery learning. The main characteristics of SimQuest are that it allows for a flexible way of authoring simulations and instruction concurrently. Using a platform for communication between simulation and instruction, called a model wrapper, SimQuest allows that an author starts at any point in the authoring process, using characteristics of the simulation model before this model has actually been implemented. This also allows different authors, for instance a modelling expert and an instructional expert, to co-operate and simultaneously develop different parts of the learning environment. The approach using the model wrapper ensures that, when brought together, the different parts of the learning environment will co-operate fluently.

The paper outlines authoring process that a SimQuest author will use and presents the current implementation of the SimQuest system.

Key words: Simulation-based learning, Authoring systems.

1. Introduction

This paper reports on SimQuest, a system for authoring simulation-based environments for discovery learning. As we have noted previously (De Jong, 1991; De Jong & Van Joolingen, 1996) simulation-based environments provide an excellent means for discovery learning, because learners can experiment freely with the simulation and discover relations in the domain. Simulation environments are safe and often cheap when compared with real system. Moreover they allow for manipulation of the time and distance scale of systems, so that processes can be made visible within an observable, reasonable time.

Discovery learning means that the learner performs a self-directed activity with the goal of an adequate understanding of the domain. This self-directedness of discovery learning implies that the learners construct their own knowledge, of which it is believed that the quality of such knowledge is higher than in the case that all domain information is presented directly to the learner in an expository way (e.g. Jonassen, 1991). However, it has often been argued that engaging learners in a discovery activity with a self standing simulation is ineffective. Discovery learning processes place a high demand on the learner with that they are not performed correctly, with the effect that learners are not able to construct much knowledge. An overview of learning processes needed for discovery is given in Njoo & De Jong (1991). The main categories they distinguish are transformative processes in which information is generated or transformed and regulative learning processes, to control the learning process itself. Transformative processes include hypothesis generation, experiment design, prediction and evaluation. Regulative processes include planning and monitoring.

In order to overcome this drawback, attempts have been made to support the learner in the discovery process (see de Jong & van Joolingen, 1996 for an overview). Among the types of learner support are: hypothesis notebooks (Shute & Glaser, 1990, Van Joolingen & De Jong, 1993), goal decomposition tools (Singly, 1990), assignments and exercises (De Jong & Van Joolingen, 1995), and model progression (White & Frederiks, 1990). Each of these 'instructional measures' attempts to support one or more learning processes needed for successful discovery. For instance hypothesis notebooks attempt to support the learning process of hypothesis generation, assignments support the planning process.

Another problem related to simulation-based discovery learning is the creation of these environments. Creating simulations requires expertise in simulation and programming. If you want
to embed these simulations in instructional measures, the systems to be created become very complex, and out of the scope of an average teacher or instructional designer. Also it is very hard to maintain systems created in high level programming languages, and to reuse material from one environment into another.

These considerations have resulted in the SMISLE system (Van Joöligen & de Jong, 1996). SMISLE is an authoring system for simulation-based discovery environments which allows authors to create these kinds of environments without having to program. SMISLE supports the author in both tasks of creating simulation models and instructional measures around them. SMISLE is organized around a number of building blocks, which represent logical parts of a learning environment. SMISLE includes building blocks for creating models, creating learner interfaces to these models, and for creating instruction. The latter include model progression, assignments, a hypothesis scratchpad and explanation. An author using SMISLE would take the building blocks templates from a library, and use a dedicated editor to tailor (specialize) the building blocks to fit in with the learning environments.

One of the strong points of SMISLE is that instruction and simulation are strongly integrated. For instance, an assignment can communicate directly to the model, for instance for setting up a specific example for the learner, to set a starting point for practice, or to modify learners' behaviour with the model. Such a strong interaction can be achieved in SMISLE by an author who does not have to write a line of programming code. Assignments and other instructional measures are created in close connection to the model and during their construction both the author and the system can draw upon information stored in the model, such as information about variables and about the model's internal structure. This allows authors to create powerful integrated learning environments.

The SMISLE approach has been tested by constructing five applications in various domains, like mechanics, electricity and plant operation (Van Joöligen and De Jong, 1996). These learning environments have in their turn been evaluated with learners (Swaan, van Joöligen, & De Jong, 1996; de Jong, Hirtel, Swaan & van Joöligen, 1996; de Jong, Martin, Zamarro, Esquembre, Swaan, & van Joöligen).

One of the drawbacks of the SMISLE approach is that it imposes a fixed order on the way the author creates a learning environment. Because instructional measures depend on many ways upon the information stored in the model, the model has to be created first. Also, model progression had to be created before anything else except the model because other instructional measures were ordered along the line of the model progression. Imposing this sequence resulted in consistent learning environments, but authors in our evaluation study felt it as being a straight-jacket. Authors often want to start with creating an interface to determine the visual appearance of a simulation before the actual model is created. Some others wanted to outline instruction before creating interfaces and models.

Also it was found that, despite the simulation tools that were included in SMISLE, for many authors the modelling task was really the bottleneck in creating applications. This is a task that they would rather give to someone else. But the problem of fixed authoring sequence would then require them to wait until the model(s) this modelling specialist would make would be ready and complete. Meanwhile the author can do nothing except making a paper design.

Both these drawbacks make that when working with SMISLE a complete design of both the model and, to a lesser extent, the instruction, should be made before starting implementation. Cooperation between authors and a flexible authoring process in which authors can develop different parts of the learning environments concurrently are not supported within SMISLE.

The remainder of this paper describes SIMQUEST, SMISLE's successor, which offers a structure to overcome the drawbacks listed above and stands as a very flexible authoring system for creating simulation-based instruction.

2. SimQuest basic structure

SIMQUEST is an authoring system for simulation-based discovery learning. SIMQUEST is the main product of the SERVIVE project. We start this section by describing the process that the author goes through in designing and implementing a SIMQUEST application. In designing SIMQUEST the main
idea was to offer a platform in which authors can carry out this process in any sensible order, to exchange material with other authors and in which they can track what they have been doing. A special role is conceived for a so-called service provider, who is capable of providing authors with new elements they may need for their learning environments.

2.1 General description of the SimQuest authoring process
The authoring process is the sequence of actions that an author working with SIMQUEST has to perform in order to build a complete learning environment. This authoring process should reflect the basic principles of the SIMQUEST system and project: flexible authoring, flexible exchange of material between the service provider and/or different authors and conceptual support for the author. In the sequence of projects that preceded the SERVIVE project (SIMULATE and SMISLE), the concept of library was introduced as the repository of building blocks, templates for using in a simulation learning environment. In De Jong and van Joolingen (1995) this lead to the description of the authoring process as: selecting a building block from a library, instantiating and specialising it, and use it in a learning environment. This basic notion is still useful for describing the authoring process in SIMQUEST.

This means that a SIMQUEST author will select elements to work with from a central resource. We will call this resource a library. A library is an object within the authoring environment which 'knows' which elements are available for the author and organises these elements in such a way that the authors can find the elements they need. Once an author has selected the library element it can be instantiated, meaning that a copy of the element is created that can be edited by an appropriate editor. Specialising the object is done by opening the appropriate editor on the element. The term specialisation will be replaced by 'editing' in the remainder of this document. SIMQUEST consists of a set of tools enabling this authoring process and a set of libraries containing the elements on which the tools can operate.

The authoring process as described above requires that authors have access to libraries, can select from them, and copy the selected elements to the learning environment they are working on. Then, appropriate editors will be used to edit the elements selected.

2.2 The SimQuest library
From the author's point of view, a library in SimQuest can be seen as a collection of elements that can be used in a learning environment. This means that the author will have access to a tool, called the library view, to browse through the library and to select elements needed. The library exists of generic templates for simulation models, instructional measures, and simulation interfaces, as well as of instantiated elements, suitable to use only in a limited range of domains. Generic templates will as a rule be provided by the SimQuest service, but authors may add their own - instantiated - library elements too.

The basic design principles require that the SIMQUEST libraries of models, instructional measures, and interface elements are not static entities but allow that elements can be added and removed in a flexible way by an author. Of course this requires a tool for authors, allowing them to select a file containing one or more new library elements, loading them into the library and from that moment on be able to work with the elements added. In this respect all library elements are of equal status, and they all respond to a basic set (or protocol) of basic operations. For instance, all library elements provide an edit operation, which will open an appropriate editor to edit the element. In an internal document these protocols have been defined.

2.3 Managing a learning environment
Each tool in the SimQuest authoring environment produces specifications of an element of a learning environment. For instance, the modelling tool generates a specification of the components of a simulation model in terms of variables and the equations to calculate the values of those variables, an assignment editor will generate a specification of an assignment, etc. Once the learning environment is started, these specifications will be used to create objects which enable the actual learning session with a learning environment, for instance, a running simulation, or an assignment which sets a state in the simulation and asks the learner to explore that state.
Since the instructional measures interact with the simulation, a protocol needs to be defined for this interaction. Moreover, the specifications of instructional objects will, as a consequence of this interaction, be dependent on the specifications of the simulation model. However, one of the principles stated in the previous section says that an author is not bound to start the authoring process with any specific component of the learning environment. This introduces the need for streamlining the dependencies of the simulation using a 'temporal' description of the simulation model, which can be created before the model itself. This object will be called the model wrapper, since it contains information on the variables present in the simulation model. The model wrapper is the main resource of information for any object that an author can create, with the exception of model specifications. Instructional measures and simulation interfaces can interrogate the model wrapper on the names and properties of the variables present. Also, they can send a request to create a new variable. Figure 1 displays the position of the model wrapper within the structure of the learning environment. It should be noted that in a typical learning environment there will be more than one model wrapper present. SimQuest supports the presence of multiple simulation models in one learning environment, in order to support the concept of model progression. Model progression is revealing the properties of the domain in a gradual way to the learner by showing models of increasing complexity (cf. White and Frederiksen, 1990). For each model present in a learning environment, there will also be a model wrapper in order to communicate with it.

Model wrappers have an important role in the authoring process. They are temporary placeholders for the information on the models they will contain. As such they are a platform of communication between all kinds of elements in a learning environment. Also, they will form a platform for communication between different co-operating authors. For instance, one author can create a model to suit a description stored in a model wrapper, another author can use the same model wrapper to design an interface. The common description stored in the model wrapper ensures that, eventually, the interface and the model will co-operate.

Elements that are stored also store a model wrapper containing a description of the models they can operate upon. When an element is loaded into a learning environment the information in the model wrapper is used to integrate the element loaded.

The model wrapper draws for part of its information upon the description of the model. Because of the openness of SIMQUEST we cannot make assumptions on the internal characteristics of the model. In principle, for each new type of model, a model wrapper is created to be able to supply the SIMQUEST environment with the information needed. Two types of information from simulation models are needed: the names and types of variables in the model and a decomposition of the model in subcomponents (if any), which can be needed by instructional measures to generate explanations.

Models may use their means of addressing variables by internal names, meaningful for a process performing the actual simulation. However, these names are not necessarily suitable to be used by the learners using the SIMQUEST learning environments. Therefore, the model wrapper will allow for using 'external names' for use by the learner and all instructional measures and 'internal names' for use internally in the model. The model wrapper will provide for the necessary links between the two name spaces.

During run time, a specialized version of the model wrapper will manage the interaction between the model, the simulation interface and the instructional measures. This runnable model wrapper 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.
A final important aspect of learning environments is that of control. Learning environments typically consist of multiple models (and their wrappers), instructional measures, and interfaces. Not all of this information can be presented at once to a learner. Therefore the notion of control is introduced. The control mechanism determines for each element in the active learning environment, when it can be 'active' and/or 'enabled'. Being active means that the element exerts some control over the learning environment to implement its goal, for instance, a simulation interface presents itself, or an assignment asks a question to the learner. Being enabled means that the learner is allowed to select the element to become active.

The SIMQUEST authoring tools will allow an author to construct a system of the structure presented in Figure 1. This means that the author can use editors to edit the specifications of the simulation model, the simulation interfaces and all instructional measures. From these specifications, an actual learning environment can be generated. The SIMQUEST authoring environment itself will be a lean environment. The real meat of both the authoring and the learning environment is in the libraries of building blocks, which provide the real functionality, including the specific editing facilities to modify the environment. The authoring environment itself is just a kernel into which new library elements can be plugged.

At any moment during the authoring process, the author may switch to 'learner mode', which will display what the learner will see. The learner will usually see a simulation interface from which the simulation can be controlled, a window showing which instructional measures are enabled as well as one or more windows through which learners may interact with instructional measures, for instance in which questions are asked and answers should be given, or in which an explanation on some phenomenon is displayed. The author can test the learning environment from within the authoring environment before distributing it to learners. When the author finishes the learning environment, it can be packed as a self-standing program. A self-standing SIMQUEST learning environment is a complete environment containing all functionality to run the simulation and instructional support created by the author, but it will not contain any functionality needed to modify the learning environment.

SIMQUEST, therefore, allows that the authoring process can be carried out in any sensible order. This is made possible by the introduction of the model wrapper as a storage of model
information independent of the actual model. Models and instruction can be developed independently of each other, and, using the information in the model wrapper, can be combined in a later phase of the authoring process.

3. SimQuest version 1

Based on this principles, a first version of SIMQUEST has been developed. In Figure 2 a screen dump of the main authoring windows is displayed. These windows include:

The author view provides the main functionalities, such as loading and saving of library and application views. It also displays the names of the library and learning environment that are currently loaded.

The library contains a list of objects or building blocks that represent either the templates from which instructional measures are created, widgets that can be imported into a simulation interface associated with the learning environment or objects from which models may be created. New objects can be added by loading them from disk. These files should be provided by the service provider. Otherwise the contents of this library window are fixed. Of course the author may store a "personal library" with building blocks created for private reuse.

The application view provides an overview of the elements that make up a learning environment. An element can be added by dragging an object from the Library view and dropping it into the learning environment view. This operation creates an instance of the building block which can then be edited using the element's editor. The application view is organized using the model wrappers it contains. When selecting a tab sheet with the name of the model wrapper on it, the instructional measures connected to this model wrapper are displayed.

The basic authoring process as described in the previous section is materialized as follows. The author selects an element in the library and drags it to the application view. When a model wrapper is already present in the application and was selected the new element is automatically connected to this model wrapper, otherwise the author has to do this later manually. If the author drags a new model wrapper a new page is created automatically. The author can then select and element and edit it. The editor that appears is tailored to the element that is selected. Using the editors, the author can fill in every detail needed for specifying and linking the elements making up the application. Figure 2 includes an editor for an assignment. The page displayed in this figure shows how for assignments an initial state can be set as a starting point for practice. Other pages, which depend on the type of assignments, allow the author to set other characteristics of the assignment. Currently Simquest supports the following types of assignment:

Investigation assignment: This is essentially a multiple choice type question where the learner has to select the correct answer from a selection of possible answers. These answer has to be found by exploring the simulation. A typical question would be to investigate the relation between two given variables. A more complex investigation assignment can be specified by asking the learner to select all the correct answers from the list of propositions given. Feedback on answers consists of explanations why the answer chosen is right or wrong. A prototype of a learner modelling module for generating feedback on the experimentation behaviour of the learner in relation to this kind of assignment has been developed and will be introduced in the SIMQUEST library soon.

Explicitation assignment: This is a variation on the investigation assignment. While the former could have an optional simulation associated with it, explicitation assignments must have one. Explicitation assignments have sets of initial states for the simulation associated with them, the role of the learner being to run the simulation with these different initial sets and to observe the impact on the simulation. These observations are then used to answer the question, which are presented in the same manner as investigation assignments. Once again, it is possible to ask the learner to select all correct answers.

Specification assignment: In this assignment the learner has to predict the values of certain variables when the associated simulation stops. The values predicted by the learner are allowed a deviation from the absolute values as specified by the author in either absolute or relative terms.
Operation assignment: Here 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 has to be reached.

These assignments are edited as independent objects, linked to a model wrapper. The SIMQUEST library provides templates for these assignments, their corresponding run-time objects are responsible for carrying out the required system actions for implementing the assignment.

The SIMQUEST authoring system provides editors for each of the elements of its libraries. Apart from the assignment editors, there are editors for simulation models, simulation interfaces and their elements, like animations and graphs, explanations, and for tests to administer to the student.

Learners interacting with a SimQuest application, will see an interface to the simulation and a windows which will tell them which kinds of instructional support are available on a certain moment. What is on this list, e.g. assignments and/or explanations, is determined by the author, by specifying for each object, as part of the editing process, when it is “enabled”. The learner can select from this list, and, for instance choose to perform an assignment. Also the author can determine that at a certain point, for instance when a certain condition in the simulation occurs, that a certain instructional measure should become active. These conditions may include the current knowledge state of the learner as inferred from interactions with the simulation model and the instructional measures. In this way, learning environments can be defined to be somewhere on the continuum between completely guided and completely free.

4. Conclusion
SIMQUEST as described in this paper provides a new step forward in the design of environments for simulation-based discovery learning. SIMQUEST provides a platform on which authors can co-
operatively and in a flexible way design learning environments based on simulation. The introduction of a model wrapper as platform for communication between, on a technical level, simulation model and instruction and, on a methodological level, different authors and different phases of instructional design, provides the flexibility needed for the complexity of the task of designing these kinds of learning environments. SIMQUEST is developed in the SERVIVE project, which includes further activities like on-line help, pedagogical advice for the authors, methodological support for authors, and evaluation studies with authors and learners. Currently a number of applications has been built with SIMQUEST in the domains of electricity and mechanics.

REFERENCES

ACKNOWLEDGEMENT
The work presented here was carried out in the SERVIVE project, which is partially funded by the European Union under contract ET1020. Partners in SERVIVE are: University of Twente, University of Amsterdam, CNPQ, RGC c/o (The Netherlands), IEC Marconi Simulation & Training SSI (UK), Ingevission, SSI (France), IPN (Germany), University of Murcia, Sallejanso Zaragoza (Spain). We gratefully acknowledge the contributions to SIMQUEST of: Roberto de Hong, Michael Kroger, David Scott, Pete Baker, Daniel Garaghan, Andre Aluse, Humberto Martinez, Miguel Calvano, Paco Ekeuembe, Ermeso Martin, Hermann Hirtel, Ian Blankenburg, Sven-Ole Falsen, Thomas Geschke, Bert Wiebersdink, Bert van Velthuijen, Jos Gerdijen, Marnix Morsink, Vincent Blhoku, Rogier van Stapelo, Mercedes Lopez, Mercedes Gutierrez, Mark Gelldevj, Koen Veermans, Janine Swaak, Renate Limbach, and Jules Pieters.