Design and Implementation of Simulation-Based Discovery Environments: The SMISLE Solution

WOUTER R. VAN JOOLINGEN AND TON DE JONG
Faculty of Educational Science and Technology
University of Twente, P.O. Box 217
7500 AE Enschede, The Netherlands

This article describes the design of interaction with the learner in simulation-based discovery environments. This interaction is by definition of a mixed initiative: In discovery learning it is the learner who takes the lead, but also the learning environment should take an active role in shaping itself on the basis of monitoring the development in the simulation and the actions by the learner. The SMISLE system, which is an authoring system for integrated simulation-based learning environments, aims at offering designers of learning environments a generic solution for defining the structure of the learning dialogue. The solution chosen is one of integrated, object-oriented design. This means that a designer can specify elements of the dialogue ("instructional measures") taken from a generic library and alter their characteristics. Moreover, the designer specifies for each instructional measure under which conditions it may become active, thus specifying control over the complete learning dialogue. The current article describes the way instruction is designed in SMISLE and presents five applications that were created using the SMISLE system.

INTRODUCTION

Learning is, nowadays, no longer seen as the more or less direct transfer of information from an "authority" (e.g., a book or a teacher) to a learner, but
as a process of knowledge construction in which learners play an active role. In the wake of this shift in view on learning and instruction, new types of learning environment have come about that require learners to discover concepts and relations in a domain. These notions have received particular attention through the introduction of computer simulations as an instructional device. Computer simulations can play an important role as a training device for reasons of safety, economy, and for social reasons (de Jong, 1991). More important, however, is that simulations are well-suited to discovery learning since they contain a model that is not directly visible for the learner, and, in order to learn about the properties of this model, the student has to engage in discovery activities.

This appropriateness for discovery learning is probably one of the reasons for the popularity of computer simulation in instruction (de Jong, van Andel, Leiblum, & Mirande, 1992). However, empirical studies did not always show the advantages of discovery environments compared to more traditional ways of instruction, in terms of effectivity of the learning process (Carlson & Andre, 1992; Rivers & Vockell, 1987; de Jong, de Hoog, & de Vries, 1993). The reason for the possible ineffectiveness of computer simulations is that learners encounter difficulties in the discovery process which they cannot overcome on their own. A number of these problems are well-documented such as the inability to formulate hypotheses (van Joolingen & de Jong, 1991; Njoo & de Jong, 1993a,b), the switching between hypotheses and experiments (Klahr & Dunbar, 1988), and the burden of regulatory processes such as planning, keeping track of what has been done, and checking (Shute & Glaser, 1990). A possible solution for these problems is to provide learners with support, in the form of "cognitive tools" (Lajoie, 1993) to assist in performing the processes of discovery, in addition to a pure simulation.

The topic of this paper is to deal with some typical problems concerning the design of supportive instructional measures for learning with computer simulations. It is argued that this support requires a new way of controlling the interaction with the learner, because the initiative for this interaction is neither completely with the learner nor completely with the system. This results in two problems: how to control the interaction with the learner during run time and how to allow an author of a simulation learning environment to define in a transparent way the aspects of this interaction. The next section describes the nature of these problems; it is followed by a section describing the solution followed in the SiMiSE project.
SIMULATION-BASED DISCOVERY ENVIRONMENT

Characteristics of Simulations

Simulation-based discovery environments differ from the more traditional learning environments in a number of ways. We concentrate on two of these differences. First, the nature of the domain model in simulation-based discovery environments is different from that in other kinds of automated instruction. For instance, in traditional computer-based training (CBT) the domain model is not explicitly present in the system, but only implicit in the information presented to the learner. In intelligent tutoring systems (ITS), the domain model often is a rule-based system, designed specifically for instruction about the domain. For simulations, however, the domain is mostly, and most naturally, modelled in terms of a quantitative, executable model of the system to be learned. Such a model usually is designed for efficient and accurate calculation of the simulation variables and normally does not have instructional features.

Second, the type of interaction of a learner with a discovery environment is radically different from the more traditional interactions with learning environments. This interaction is not a system-initiated dialogue with the learner, as in traditional CBT or dialogue-based ITS, where the system presents information and the learner reacts to this information, for example, by answering questions. Also, the initiative of the interaction is not completely with the learner as in database applications. Instead, both system and learner may initiate a sequence of events. Learners can manipulate the simulation at their own initiative, but, once started, the simulation is continuing independently of the learner, whether manipulated or not, and events resulting from this ongoing simulation may result in learning opportunities.

These two characteristics of simulation-based environments put specific requirements on the way cognitive tools for instructional support of discovery learning with simulations are designed. A crucial issue here is to find a balance between guidance and learner freedom (Elsom-Cook, 1993).

Handling Learner Freedom

Simulation models are designed to calculate the values of certain output variables on the basis of values of input variables, and, in the case of dynamic simulations, also on the basis of time. In a simulation-based
learning environment, a learner can manipulate the inputs before or during run time of a simulation and can see the results of these manipulations in terms of outputs. As long as there is no other component in the system, this does not pose real problems, because the learner can concentrate on the single simulation and the system is only reactive to two events: change of input by the learner and progression of time. These two events do not compete with each other for priority.

In guided discovery environments, however, the learning environment needs to be aware of the actions of the learner and of the development of the simulation in time and over different simulation runs. For example, the learning environment may want to ensure that the learner sees a number of specific states in the simulation, for instance, critical events. In such a case, the learning environment must act as a kind of watchdog over the learner's actions and over the simulation and register if such critical situations have taken place. Information of this watchdog activity may then be used for activating instructional measures which can support the learning process.

In their turn, the instructional measures need to be aware of the state of the simulation and the state of other instructional measures of which they may be dependent. As the interaction between the learner and the system is of mixed initiative, each of these states may be changed at any moment, by a learner action or by the occurrence of a specific event in the simulation. Also, instructional measures should be able to act as an agent in the learning environment itself, by changing the state of the simulation and/or communicate with the learner.

As an example, we consider a learner involved in a discovery environment on the physics topic of oscillatory motion. Typically, the learning goal of such an environment is that the learner discovers the main relations that underlie the physics of this type of motion. Examples of these relations include the effects of the strength of the spring on the frequency of this motion and the influence of friction on both the frequency and the amplitude of motion. In an unguided situation, learners would be free to vary all available input variables, like the friction coefficient and the strength of the spring, and be able to observe all available outputs, like the frequency and amplitude. Alternatively, in a completely guided situation, learners would probably go through a fixed sequence of experiments with the simulation. These experiments would be defined in terms of sets of values for input variables; relevant output variables are then plotted in a graph or listed in a table.

Both situations have their merit. In the free situation, learners are in control, which is assumed to be beneficial for the development of
experimentation skills as well for triggering learning processes which result in a deeper understanding of the domain. The drawback is that the learning environment has no control over the situation in case learners start to flounder, for instance, when they start to give meaningless values to the spring constant or start to perform experiments in which many variables are manipulated at the same time. This is no problem in the complete guided situation, but of course that situation leaves absolutely no control to the learner.

What is needed in the case of guided discovery is to monitor the behaviour of learners and to intervene only when necessary. As noted above, this poses problems due to the nature of simulations. If, for instance, in the example of oscillatory motion, the learner chooses values for the strength of the spring which cause breakdown of the system, we would like a guidance system to be able to use these situations for explaining the concept of resonance (which causes the breakdown). An attempt for doing this would be to calculate the values for input variables for which this phenomenon takes place and then react when the learner chooses these values, and subsequently react with a (system-generated) explanation. This approach falls short for two reasons. The first is that the phenomenon takes place only after a certain amount of (simulated) time. This means that an immediate reaction of the learning environment is too early for the learner, and it is difficult to determine the right time to come up with the explanation. The learning environment depends heavily on domain knowledge to determine the right moment of intervention. The second reason is that learners run the risk of never seeing such events, because the width of the critical region for input variables is small. Therefore, it is better in this case that the learning environment diagnoses the state of the output variables of the simulation and reacts when these indicate the occurrence of instructionally interesting events. Also, the learning environment needs to have the possibility to take control over the inputs to set the simulation in a position to take the learner to an interesting event.

The example given above shows that, for fruitful learning, a learning environment around a simulation needs to have control over the simulation environment. The control over the learning environment is needed to offer various kinds of instructional support, which include:

- decomposing instructional goals. In simulations of complex systems, care should be taken that the learner does not try to grasp the situation all at once. Therefore, the learning environment must try to set a chain of goals for the learner in order to come to a full understanding;
focusing attention. If critical or interesting events occur in the simulation, the learner should be made aware of this by pointing to the values of relevant outputs.

For implementing these supporting functions, the learning environment needs to monitor the behaviour of the learner and take control over the simulation when necessary. This means that the learning environment needs the following “handles” on the simulation model:

- control over the input variables. For instance, this can be used to draw the learner’s attention to certain critical and/or interesting events in the simulation;
- access to output variables, which can be used to determine if certain interesting events take place; and,
- control over the presentation of the simulation, which allows to determine which variables the learner can or cannot observe and/or manipulate.

If we were only designing a simulation program for harmonic oscillations or any other specific purpose, it would be relatively easy to implement a system which allowed for this set of controls over the simulation model. The simulation would be implemented in a programming language like C++, and all necessary monitoring and control would be hard-wired in this simulation program. Actually, a number of simulations of this kind already exist, like the Cardiac Tutor, (Eliot & Woolf, 1995), QUEST (White & Frederiksen, 1990), ELAB (Böcker, Herczeg, & Herczeg, 1989), Mach III (Kurland & Tenney, 1988), Smithstown (Shute, Glaser, & Raghavan, 1989), Voltville (Schauble, Glaser, Raghavan, & Reiner, 1991), and Re-fract (Reimann, 1989), to mention a few.

In the current paper, a different stance taken in the SMISLE (System for Multimedia Integrated Simulation Learning Environments) project is illustrated. The SMISLE project aimed at designing an authoring system for simulation learning environments. This means that instead of finding a domain and context specific solution, as is hard-wiring the control in the simulation, a generic solution for the design of simulation-based learning environments was sought. This solution should be usable by domain experts who have no specific programming experience. The goal of the SMISLE system is to provide an environment in which the simulation model and the instructional support around it are developed in an integrated way, which means that the author should be able to develop the simulation model and instructional support in such a way that the control over the
simulation by the instructional support is ensured in a consistent way. Existing authoring systems for simulations are RIDES, and its predecessors (Munro, 1993) and PowerSim (Vavik, 1993) either require programming knowledge or do not integrate instruction with the simulation model. The SMISLE system offers a solution for embedding a simulation in instructional support in an integrated way.

THE SMISLE AUTHORING SYSTEM

Instructional Measures in SMISLE

The SMISLE authoring system (de Jong et al., 1994; de Jong and van Joolingen, 1995) aims at providing domain experts with a set of tools for designing simulation-based learning environments. These learning environments provide learners with simulation and support them in their discovery process by means of a number of instructional measures. The aim of instructional measures is to enhance the quality of the discovery learning processes. Therefore, they should not significantly reduce the freedom of the learner while discovering the simulation. SMISLE provides four types of instructional support (de Jong et al., 1994; de Jong et al, 1996):

1. Model progression. A learning environment created with SMISLE may contain a number of different simulation models, ordered along dimensions like difficulty, "order" (qualitative vs. quantitative), or perspective on the domain (see also White & Frederiksen, 1990). This instructional measure takes control over the input variables in a global way: It determines which variable are present and visible for the learner at a certain time.

2. Assignments. Assignments provide the learner with short-term goals, like finding a specified relation, predicting the behaviour of the simulation, or achieving a specified simulation state. In cooperation with model progression, assignments decompose the overall learning goal of a simulation in a number of subgoals. Assignments can take further control of input variables in two ways. They can block the access of one or more inputs and they define an initial state for all inputs. In this way assignments can set up interesting events. Also two types of assignments allow for control over the output variables. By defining conditions for these outputs, for example, an amplitude which grows above a certain threshold, assignments can select interesting events in the simulation.
Finally, when activated, assignments can also select a specific screen representation for the simulation.

3. **Hypothesis scratchpads.** SMISLE provides a tool for learners to note their hypotheses on the simulation and simulated domain. Hypothesis scratchpads provide the learner with a structure to state hypotheses using variables and relations from the simulation and relations defined by the author. The hypothesis scratchpad was introduced previously in van Joolingen and de Jong (1991, 1993), based on ideas by Shute, Glaser, and Raghavan (1989).

4. **Explanations.** The SMISLE system can generate explanations on various aspects of the simulation. For example, it can generate a causality chain, or it can provide a structural overview of the simulated system. Also, the author can define textual, graphical, and multimedia explanations. These explanations can be used to provide for extra information on variables, relations, or events in the simulation.

![Figure 1](image-url)

Figure 1. A typical SMISLE learning environment (SETCOM, a simulation of oscillations). The simulation window, top left, is surrounded by a number of windows offering the learner instructional support.

Figure 1 shows a typical SMISLE learning environment with a window (top left) giving access to the simulation and four windows giving access to instructional measures. In the design process the author creates a
number of these instructional measures and specifies the dependencies of the instructional measures on each other, of learner responses, and of the simulation. The author does this in an integrated, object-oriented way.

Integrated Object-Oriented Design of Instruction

A SMISLE author has to define one or more simulation models of the domain together with the instructional environment around it. This is accomplished in an integrated and object-oriented way. The authoring process is integrated because the various elements of a learning environment draw upon each other for information and are available during the authoring process. As soon as one part of the learning environment is modified by the author, other parts can use the changed information directly. For instance, all instructional measures have knowledge of the names and types of variables in the simulation. As soon as the simulation is modified by the author the changes are propagated to the instructional measures. Instructional measures update themselves automatically to this changing information. For example, the hypothesis scratchpad updates the lists of variables which learners can use in their hypotheses. SMISLE ensures that the information offered on the scratchpad is always consistent with the simulation.

The authoring process in SMISLE is also object-oriented, meaning that the author can concentrate the authoring activity on one element of the learning environment at a time. The author selects the simulation elements and instructional measures from dedicated libraries and adapts them to the specific situation of the learning environment under construction.

Typically, an author starts with the definition of the simulation, which is the core of the simulation environment. For this, SMISLE contains a functional block editor in which blocks from the library, which are small models in itself, can be combined to form larger models (for a detailed description see Scott, 1993). The next step in the process is the creation of the learner interface to the simulation by selecting predefined interface elements, like sliders, graphs, and numeric controls, and linking these elements to input and output variables in the model. Once the model and the simulation interface are ready in a first version, the instructional measures in the learning environment can be created and connected to the rest of the environment. The design process is not linear as this sequential description may suggest: The author can always step back in the process to modify the results of previous steps.
Designing a SMISLE Instructional Measure. A SMISLE instructional measure is an object capable of interacting both with the learner and with the simulation. For instance, an assignment communicates with the learner on the goal of the assignment and on the result of the learners actions, that is, if the goal has been achieved or not. Also, it is capable of setting the simulation in specific states and to check with the running simulation whether goals are achieved or not, for example, if a prediction given by the learner is correct.

An example of an assignment is the optimisation assignment. This type of assignment sets the learner the goal to find the optimum value of some output variables given, for instance, to find the situation of critical damping in oscillations. An optimisation assignment is defined in terms of targets (the optimum values) and constraints, which should be kept during the optimisation process. When an optimisation assignment is activated, the constraints become effective. In the event that one of the constraints is broken, the simulation is halted and control is passed back to the assignment object, which takes appropriate action. Such an action can be, for example, setting up the simulation for a new attempt or issuing an explanation on the causes of breaking the constraint. Figure 2 shows the editor for optimisation assignments from the SMISLE system. In SMISLE, each type of instructional measure has its own dedicated editor.

Assignments and other instructional measures are self-contained objects. They know for themselves what to do in various situations once they are activated. For instance, they can control the access the learner has to variables in the simulation.

SMISLE does not have an object defining an overall instructional strategy, which decides when an instructional measure may become active. The object-oriented approach in SMISLE requires that each instructional measure is able to determine that itself. This means that there is no generic control facility keeping lists of instructional measures and deciding globally which instructional measure to activate. Instead, all instructional measures in SMISLE have enabling conditions, edited by the author, specifying when the instructional measure may become active. These conditions are defined in terms of the state of the simulation and other instructional measures. For example, an assignment has the following possible states: (a) enabled, (b) active, (c) succeeded, (d) failed, and (e) completed. The enabling condition of an assignment can be defined as, for instance: “assignment_2 completed AND assignment_3 completed.” The SMISLE system provides a graphical editor for enabling conditions (see Figure 3). This way of defining the control over instructional measures has the advantage
that the control is adapted easily. For instance, an instructional measure is inserted without having to consider the exact place in a control structure. Also, instruction is ensured to be very flexible. There is no need to set global rules for instruction or to predefine sequences of instructional measures. Instructional measures are enabled and disabled as they are, without global control.

Figure 2. The SMISLE editor for an optimisation assignment

Instructional measures also have conditions which determine when they are actually activated. One way of activating an instructional measure is selection by the learner when enabled, but instructional measures may also become active on their own initiative; for example, an explanation may pop up at the moment the learner fails to complete an assignment.

*Designing the interaction with the simulation.* In a SMISLE learning environment, instructional measures are constantly aware of the state of the simulation, and, in its turn, the simulation is aware of instructional measures monitoring its activity. For instance, a model progression level, which is an instructional measure, can hide certain variables for the learner. Assignments can set the simulation in a specific state (specified by the
author), limit the learner's control of input variables, and put constraints and targets on output variables. The simulation then checks these constraints and passes back control to the assignment as soon as a constraint is broken or a target reached. The assignment reacts by handling the event and by modifying its state.

Each time an instructional measure modifies its state, a waterfall of events takes place. All other instructional measures whose state depends on the changing instructional measure will reevaluate their enabling and triggering conditions, resulting in new changes of states. Effectively, this makes the collection of instructional measures a forward chaining rule-based system. For instance, if a learner succeeds with an assignment, other, more difficult assignments may become enabled. Also, on failure of an assignment, extra explanations or a remedial assignment may come up to help the learner proceed in the learning process.

This updating mechanism ensures that the state of the learning environment is kept consistent at all times. At any moment, each instructional measure knows if it is enabled or not, or if it is being activated. This means that the interaction with the learner is controlled in a distributed way in which each instructional measure handles a part of the dialogue with the learner.

SMILE APPLICATIONS

In the current section, five applications that were created with the SMILE authoring system are reviewed. After a short general description of each application, the authors discuss the way the SMILE control mechanism is used to determine the level of guidance through the instructional support available to the learner.

SETCOM. SETCOM (van Joolingan, van der Hulst, Swaak, & de Jong, 1995) addresses one-dimensional oscillatory motion. SETCOM implements simple to complex model progression, with three different levels. The levels are, in increasing order of complexity: simple harmonic motion, damped oscillations, and forced motion. Learners typically go through these levels in this order. Figure 1 displays a screen image of the SETCOM environment.
SETCOM contains a number of assignments. Most of the assignments are investigation assignments which ask the learner to investigate the relation between two given variables, for instance, find the relation between the force constant and the frequency. SETCOM also contains optimisation and specification assignments. The goal of an optimisation assignment is to control the simulation in such a way that an optimum value for an output variable is reached. Specification assignments ask the learner to predict the value of certain variables in a given situation. In addition, SETCOM includes explanations and a hypothesis scratchpad.

Guidance of the learner through the available instructional support concentrates on moving through the levels of model progression. The learner is allowed to move to the next level of model progression when all...
investigation assignments associated to that level have been completed. This means that the conditions determining the completion of a model progression are set to expressions like "investigation_assignment_1 completed AND investigation_assignment_2 completed AND ...." The order of choosing assignments *within* a model progression level is left to the learner, that is, their enabling conditions only depend on the associated model progression level. Explanations are available whenever the topic explained (usually variables) is present in the current model progression level. We call this a *controlled* method of model progression (de Jong et al., 1996) because the learning environment effectively has control over the path of the learner through the model progression. This way of control ensures that learners have shown some competency, by answering assignments, as they proceed through the progression of models.

![Diagram](image)

*Figure 4. Simulation window for transmission processes on a double line in TcEl*
Collision. In Collision (de Jong, Martin, Zamarro, Esquembre, Swaak, & van Joolingen, 1995), the topic is one dimensional elastic collisions. Processes of this kind are very common in nature; for instance, collisions between billiard balls, shooting a bullet, and the scattering of an electron by the nucleus of an atom can be treated as collision processes. The collision application consists of five model progression levels, which move from an introduction in kinematics, via introductions in conservation of momentum and energy to collisions. The final two levels offer different viewpoints on the total process of collisions and do not build upon each other, as is the case for the first three levels.

The model progression levels are ordered, with the exception of the top two levels which become available simultaneously. On each level the learner is free to choose any from a number of available assignments and explanations. In collision the choice was made to enforce this ordering by controlling the time spent by the learner on this model progression. The status of a model progression level is made dependent on the time past since the model progression level became active, thus ensuring that the learner at least spends the specified time on this model progression level. Within this time the learner is not allowed to move on to the next level, but may interact with the simulation or choose any assignment or explanation. In this way the order in which the learner accesses the various model progression levels (and the assignments and explanations that go with them) is determined by the control mechanism; also, the pace of model progression is in the hands of the learning environment. However, there is no strict control over the actual activity of the learner at each model progression level. We call this way of control timed.

TeEl. In TeEl (de Jong, Härterl, Swaak, & van Joolingen, 1996), based on the numerical solution of the telegraph equations, the transmission of changes in voltage and current are simulated and visualised in real time (see Figure 4). The instructional goal is that learners discover the basic nature of the concepts “transmission,” “superposition,” and “reflection,” and learn how to apply the model of travelling waves in order to explain transmission phenomena.
TeEl implements four levels of model progression. As the model progression proceeds, the learner gets control over more variables. Like in the other applications, at each model progression level a number of assignments and explanations are offered. Control over model progression is more in the hands of the learner than in SETCOM and Collision. Like in these applications, the model progression levels are ordered; it is a good idea to choose them in a specific order. However, TeEl gives the learner control over the pace in which the model progression levels are visited. At any point during the interaction the learner may choose to move to the next model progression level. That is, learners can indicate when they think they are ready to move to the next model progression level. Within the model progression, learners are free to choose the assignments and explanations they want, with the remark that assignments are numbered, so an order of assignment selection is suggested by the learning environment. This way of control over model progression is labelled *learner controlled*.

HPU. HPU (de Jong, Swaak, Scott, & Brough, 1995) is an environment for training operators and engineers for performing the start-up procedure of a part (the Hydrogen Purification Unit) of an ethylene plant of Exxon Chemical (see Figure 5). As opposed to the *conceptual* models underlying the applications...
Figure 6. The simulation interfaces for the four model progression levels in AUTOSTAMPING

discussed above, HPU is created to convey operational knowledge, that is, knowledge on how to operate the plant during start-up time. The instructional support in HPU is centred around a set of normal operation assignments, each inviting the learner to carry out the procedure in a specific way. The operator can select these assignments in a specific order, where the first assignment uses the procedure as described in the manual, and the last uses an expert procedure which violates some of the prescriptions in the manual, but results in a much faster start-up. With the sequence of assignments offered, learners can gradually move from what they know (the manual) to the expert procedure.

Model progression is used in HPU only to offer learners some initial practice of one of the components of the complete system. The first level conveys a model of one of a type of valve used in the simulated system, whereas the second model progression level introduces the complete system. It is at this second model progression level where the sequence of assignments takes place.
The order in which assignments are executed is determined by the learning environment. Only after one assignment has been completed, the next assignment can be selected. This control can be labelled as guided. This is the most constrained type of control; the learner is presented one instructional measure at a time.

AUTOSTAMPING. AUTOSTAMPING (van Jooldingen, Valent, Scott, & Brough, 1995) allows engineers to investigate several properties of models involved within the simulation of stamping metal sheets. AUTOSTAMPING contains several model “progression levels,” which offer different situations that are of interest to the stamping process, as indicated in Figure 6. There is no implicit order in the various levels, so learners can pick them in any order and switch back and forth between levels. This type control is labelled free and is on the opposite side of the spectrum of learner freedom as the guided control used in HPU.

Evaluation Studies

Three of the learning environments that were presented above have been evaluated empirically: SETCOM (van Jooldingen, van der Hulst, Swaak, & de Jong, 1995; Swaak, van Jooldingen, & de Jong 1996), Collision (de Jong, Martin, Zamarro, Esquembre, Swaak, & van Jooldingen, 1995) and TeEl (de Jong, Hartel, Swaak, & van Jooldingen, 1996). In these studies, the complete versions of the learning environments were compared with versions in which parts of, or all of the instructional support was left out. Tests were administered for “definitional knowledge” and “intuitive knowledge.” The overall result is that simulation-based discovery learning leads to an increase in both definitional and intuitive knowledge but especially of intuitive knowledge. Instructional support helps to increase these knowledge gains.

DISCUSSION

In this paper the authors introduced the solution taken in the SMISLE project for handling the typical difficulties of interacting with a learner in a guided discovery environment. These difficulties concern the specific type
of learner interaction, its mixed initiative, as well as the difficulty for authors to control this interaction in a manageable way. The specific, mixed initiative, learner interaction is needed to be able to offer the learner guidance and support during the exploration of the simulation model and, at the same time, retain the learner freedom that is essential in discovery learning. In the solution that is employed by the SMISLE system, the various instructional measures and the simulation are all agents of equal importance, drawing on information stored in each other. The enabling conditions, which hold for the instructional measure, control the freedom of the learner. The solution presented offers opportunities for controlling the balance between guidance and free exploration (Elsoom-Cook, 1993). In a sense the learner and the instructional measures are both actors on the state of the complete learning environment. Since the instructional measures can adapt their behaviour with their enabling conditions, the learner freedom can easily be tuned to the actual behaviour of the learner.

The four types of instructional measures defined in SMISLE allow the author to exert a number of types of control over the learning environment: control over inputs, outputs, and the screen representation of the simulation. This allows the author to specify that the behaviour of both the learner and the simulation is monitored and opens the possibility for decomposing the overall learning goals of the simulation environment and focusing the attention of the learners to interesting events in the simulation.

The author of the instructional environment is offered an integrated solution for specifying the control over the learning environment. The authoring task is supported in a number of ways. First, the presence of a library of instructional measures allows the author to concentrate on the essentials of the instructional design task; there is no need to specify the detailed behaviour of the instructional support. Second, the author can edit the various instructional measures locally, without having to pay attention to all kinds of side effects. Typically, instructional measures only depend on a few other objects in the learning environment. The author can easily streamline the flow through the instructional support by locally setting a number of conditions.

A simulation learning environment created with SMISLE combines the intelligence of two actors: the learning environment, that is the collection of all instructional objects, and the learner. The learning environment uses its rules to determine the space the learner can explore. This space consists of the part of the simulation that is available to the learner, as determined by instructional measures, and a collection of instructional measures (e.g., model progression, assignments, and explanations) to choose from.
Learners in their turn are free to make any choice within this space of exploration.

The applications created with SMISLE, as discussed in this article, demonstrate some of the possible arrangements of instruction within SMISLE. Five types of control were introduced, in increasing order of learner freedom: guided, controlled, timed, learner controlled, and free (see Figure 7). These labels only serve to indicate some gross categories of control types and illustrate various points on the continuum between complete guidance and complete freedom. The SMISLE control mechanism allows to regulate the control at a much higher level of detail than can be indicated by these terms; even the level of guidance can be changed in the course of a session with the learning environment. For each instructional measure, the moment it becomes active can be controlled individually.

![Figure 7. The SMISLE learning environments positioned on the continuum of increasing learner freedom](image)

For designing simulation-based instruction, this implies that the author can concentrate on the role a specific instructional measure plays in the learning environment. For instance, an assignment or explanation that is created as a remedy of a certain misconception can be triggered by signs that this misconception occurs, like an error in answering an assignment. An instructional measure designed to take the learner to a new part of the domain can be triggered on signs that the learner has acquired sufficient knowledge to understand that the new part, like the completion of investigation assignments in SETCOM, makes available a new level of model progression.

An interesting feature of the SMISLE control mechanism is that it can provide a hook for a (modest) way of learner modelling. Just like any other
object in the learning environment, a learner model can set control states, which can trigger or disable instructional objects. The learner model can, for instance, base these states on the interaction of the learner with the simulation, but also on the states of instructional objects themselves, like the results of assignments or hypotheses entered on the hypothesis scratchpad. In this way, the mixed initiative learning dialogue, which is typical for discovery environments and especially simulations, can be further refined.

References


