Interfaces for instructional use of simulations

Robert de Hoog

University of Amsterdam, Department of Social Science Informatics

Ton de Jong

Eindhoven University of Technology, Department of Philosophy and Social Sciences

&

University of Amsterdam, Department of Social Science Informatics

Frits de Vries

University of Amsterdam, Department of Social Science Informatics

The learner interface is the component of an instructional system that mediates between a learner and the system. Two fundamentally different approaches for interfaces can be distinguished: conversational metaphor and direct manipulation metaphor. Interfaces in both metaphors can be scaled on a dimension indicating the 'distance' between the user's intentions and the physical expression. In combining the dichotomy small and large distance with the conversational and direct manipulation dichotomy, four different interface types result. These 4 types can be applied to both the input and the output side of the interaction. Combining these yields a (4x4) 16 cell 'space of interaction' matrix. This matrix is used as a background for the rest of the paper.

We will distinguish three generic entities in the interface for instructional simulations: the model entity, the learning entity and the control entity. The model entity is further subdivided into an output and an input aspect, respectively covering the domain model and learner activity. The learning entity consists of an instructional aspect and a learning process aspect. The first one is related to instructional goals and the latter comprises everything that is related to the learning process of the learner. The control entity is mainly for high level control of the simulation environment, giving the learner the opportunity to quit, save and sequence.

All aspects of simulation learning environments have to be integrated on one screen. An attempt is made to define generic action and object classes which can be used for this ordering of input and output. Finally, we will give a brief summary of desirable hardware properties.

1. Introduction

Learning with computer simulations is generally

seen as an effective way of gathering knowledge. The main reason for this is that computer simulations solicit *exploratory* or *scientific discovery* learning. On the other hand it is clear that exploratory learning is difficult and learners may need all kinds of *support* in the learning process. This support can be supplied by a computer environment and in this respect the concept of an *Intelligent Simulation Learning Environment* (ISLE) has been introduced (see de Jong, this volume).

In de Jong (this volume) four *themes* that characterise instructional use of simulations were listed: an 'active' domain model, the presence of learning goals, the elicitation of exploratory learning processes, and the presence of learner activity. Moreover (following 'classical' literature on ITS design), an ISLE was defined as consisting of four *design components*: the domain, learner, instruction, and interface components.

The learner interface is the 'front end' of the instructional simulation, and as such it incorporates and supports all of its different aspects, and has to take into account all possible combinations and interactions of the themes and components mentioned. The study of these themes and components and the interaction between them will constitute the major part of this paper. The outline is as follows. First, we will give a brief overview of the development of man-computer interfaces during the last twenty years. This will lead to a classification of types of interfaces that can serve as a point of reference for discussing specific properties and requirements for interfaces for simulation. The next three sections will give a systematic treatment of three generic entities that constitute an interface for instructional simulations: the model entity, the learning entity and the control entity. These entities subsume the four themes that characterise instructional simulations and that were mentioned above. The remaining sections will describe general design considerations, and hard and software requirements.

2. Man-computer interfaces: an overview

An overview of the development of user interfaces during the last twenty years shows that two metaphors have governed the design of interfaces: the

conversational and the *direct manipulation* metaphors. Based on characteristics of these metaphors we present a classification of interfaces that will serve as a reference for the following sections¹.

2.1. *Metaphors for interface design*

In the mid sixties an important change occurred in the way people interacted with computers: from an overwhelmingly batch oriented way to direct interaction through a screen and keyboard. However, the communication language did not change with the introduction of interactive human-computer communication; the user simply felt more 'in touch' with the computer using keyboard and screen. The dramatic reduction in turnaround time compensated for the time needed to use the same awkward command language as in batch systems. At the same time the transition from cards and listings to keyboard and immediate responses on the screen was a frightening experience for many regular and novice users. Touching the wrong key could cause a mess on the screen, files lost, waiting for responses, red lights flashing etc. This however did not inhibit the steady growth of interactive computer use, a process speeded up considerably by the appearance of the micro-computer and the PC.

However, the computer language still remained the one used in the fifties, though now used in a conversational metaphor: man and computer are talking partners. Within this framework there has been a slow expansion of human-computer interaction possibilities. The main direction of development has been an attempt to ease the burden of adapting to the computer language for (casual) users. One of the results is the emergence of natural language interfaces. Another is the introduction of a visual component in the user interface, which is reflected in direct manipulation interfaces.

In the next two sections we will describe in more detail the two main metaphors that dominate the

user-interface landscape at this moment.

2.2. *The conversational metaphor*

Initially, communication was dominated by the language of the computer. Users had to learn its complicated syntax and semantics. This complexity can be seen as a natural consequence of the way computers were programmed for executing a task. Therefore this style of interaction is called a formal interaction style. A formal interaction is a conversation "in which the activities and data structures within the computer are presented externally in a direct representation with the minimum syntactic sugar necessary to aid human recognition" (Gaines & Shaw, 1986b, p. 102). Examples of this formal conversation style are: command languages, query languages, form fill-in systems, menu selection systems (Gaines & Shaw, 1986a,b; Shneiderman, 1987).

Several developments in hardware and software made it possible to approximate more and more to ordinary human language. Thus the burden of learning and remembering the syntax of the interaction language is relieved. This kind of communication is generally called a natural language dialogue. The human use of language is simulated by the computer and an anthropomorphic machine is created (Gaines & Shaw, 1986a,b; Shneiderman, 1987). Natural language interfacing is greatly aided by Artificial Intelligence research and logic programming languages like Prolog. With research and experiments still going on, there already exists a system "with a vocabulary of 20.000 words, 98 % of the typical English speaking vocabulary". The machine can even interpret phonetically abstruse phrases like "Write ms. Wright a letter right away" (Foley, 1987, p. 66). Recent developments are also the 'talking head' computer interface, where the computer acts as a kind of television screen: you actually see and hear someone talking from a window in the application. Through this integration of hypermedia, simulation, and artificial intelligence, human language is becoming part of the computer (Gasper, 1988). However, Shneiderman points out some problems with natural language interfaces. One drawback is the missing information (syntax,

¹The discussion will be a general one, implying that we will use the term 'user' instead of 'learner', which will be employed in the other sections.

semantics) about the objects and actions in the task domain (Shneiderman, 1987).

2.3. *The direct manipulation metaphor*

While efforts to make the computer understand as much as possible from conversational input are continuing, a new way of communicating with the computer has emerged, without typed or spoken language. This way of communicating looks more like the way man communicates with his environment: through objects and actions. It is called the direct manipulation metaphor (Hudson, 1987)².

Shneiderman coined the term 'direct manipulation' and its key ideas are: "visibility of objects and actions of interest, rapid reversible incremental actions, replacement of command language syntax by direct manipulation of objects" (Shneiderman, 1987, p. 180). Objects on the screen are representations of real world objects and actions in the most simple form are touching and moving with a special pointing device, the mouse. This means that a user can execute a task with a minimum of syntactic knowledge and only a little semantic knowledge of the internal computer language. The user can concentrate fully on the semantics of the objects and actions of the task (Shneiderman, 1987). The most spectacular direct manipulation application is the interface with which reality can be simulated and an actual direct manipulation is made possible: the user sees his hand, also simulated, manipulating something in the artificial world. The real hand wears a glove with special electronic equipment translating movements into signals, with which the computer projects a simulated hand on a head-mounted monitor. The movements of the real hand are made visible on the monitor. The special glove can also be used for remote manipulation in a place where people cannot go without risks, e.g. a nuclear plant.

The value of these types of artificial realities for simulated learning has yet to be established. Foley gives a clear description of how user and computer can act in a shared simulated world (Foley, 1987, pp. 62-67). Shneiderman suggests that research on problem-solving and learning shows the positive influence of visualisation on the learning process in comparison with textual and numerical representation (Shneiderman, 1987, p. 198). Problems lie in finding a transferable visual representation of the object from model or reality (Shneiderman, 1987). The 'desktop metaphor' used by the Macintosh user interface is a fairly easy analogy, but what about visualising complex logical constructs such as 'integrity rules' one encounters with databases?

2.4. *New directions*

In the near future, working with computers and programming will be more and more a multi-user and multi-tasking undertaking (the *distributed* use of information technology as it is called nowadays). Personal computers are integrated in networks with other computers, for instance public databases, but also with other media such as video-text systems and mixed media like Hypertext (Fiderio, 1988). The user interface is likely to be more like a workstation A3 screen with a number of windows, some with text, some with objects to be manipulated and a private selection of non-standard input and output peripheral devices. The new user interface metaphor is challenging: concurrency, communication and synchronisation of the human-computer interaction (Hill, 1987).

Summarizing, we can say that there are two major metaphors that drive the design of man-computer interfaces: the conversation metaphor and the direct manipulation metaphor. In the past, most interfaces have relied on the conversation metaphor, but during the last five years there is an upsurge in the development of direct manipulation interfaces. This change in the use of a metaphor is facilitated by relatively recent advances in technology. Multi-user and multi-tasking user interfaces are at present mostly based on both metaphors.

²One could argue that in spite of all its proclaimed advantages and superiority this development is a step back. Direct manipulation of objects is definitely not the way intelligent partners tend to communicate, they talk in some kind of language. From this angle direct manipulation is just a way of representing the interaction between dumb, inanimate objects and humans.

3. A classification of interfaces

In this section we present the elements of a classification schema for interfaces. This schema is composed from a *space of interfaces* axis and a *space of interactions* axis.

3.1. A space of interfaces

The contrast between the conversational metaphor and the direct manipulation metaphor is elaborated by Hutchins, Hollan and Norman (1986) when they examine the basis underlying direct manipulation systems. They do not try to quantify direct manipulation because there is a 'feeling of directness' involved in these systems and this feeling is difficult to quantify. The stronger this feeling, the more the user will feel familiar with a particular system. Moreover, Hutchins et al. identify two aspects of 'directness': 'distance' and 'engagement'. They construct a two dimensional 'space of interfaces', with each aspect on a dimension. A concise description of these dimensions is given below, as well as a representation of the space of interfaces.

Distance

In terms of distance there is a 'semantic distance' between the user's goal(s) and the meaning of a selected expression in the interface language (both for input and output), and an 'articulatory distance' between this meaning and the chosen physical form, also for both input and output. The conversational form uses an explicit expression, whereas direct manipulation, by definition working without explicit language symbols, uses an implicit expression. These implicit expressions are hidden in the rules for the operation of the input devices (e.g. is a click or double click required) and interpreting the output expression (e.g. different pictures or icons).

Engagement

In terms of engagement, Hutchins et al. introduce the distinction between the 'interface as a conversation' and the 'interface as a model world'. On the conversation side there is the use of some kind of language as an intermediary in the communication between human and machine. On the model world side there is no explicit language, but the 'feeling'

of direct involvement through the use of devices. One of the requirements for this feeling is the 'inter-referential' relation between input and output. This means the possibility of an output expression to be used as (part of) a following input expression. This is self-evident when ordinary text in an editor is the object of manipulation. Schema 1 summarizes these distinctions.

Engagement	Distance from user goals	
	<i>small</i>	<i>large</i>
<i>Interface as conversation</i>	high level languages	low level languages
<i>Interface as model world</i>	high level world	low level world

Schema 1 *A space of interfaces*

Locating existing interface types in this space can only be done by making assumptions about who the users are. In other words the space cannot be filled in an absolute way. If we deal for example with expert users in an assembly language, then there is already a small distance to the user goals in a low level language. Average users on the other hand, experience a large distance from their goals when working with a low level language like an assembly language.

Assuming an average, infrequent user, one could fill the cells in the matrix of Schema 1 as follows:

high level languages:

natural language systems, an understanding/talking machine in pure form, synthesizing speech, speech interpreters.

low level languages:

first generation languages, machine language in the extreme, all other kinds of languages and packages.

low level world:

peripheral devices with directions for use, mice, joysticks, scratch-pads, icons, spreadsheets.

high level world:

devices that need no instruction, direct manipulation

in a pure form, real world objects or representations.

Almost all existing types of interaction can thus be placed somewhere in the space.

3.2. *A space of user interface interactions*

The two general characteristics of user interfaces (distance and engagement) are unfolded in a two-dimensional space by Hutchins, Hollan and Norman (1986). These characteristics suggest that user input and computer output in general will have the *same* distance(s) and the same type of engagement. However, in reality, interfaces do not always fit in this simple space, because different distances and engagement types are used for input and output processes. Existing interaction processes can be better placed in a schema where input/output combinations for different entries in the space of interfaces are combined. In Schema 2 the space of interfaces is split into an input part and an output part.

This distinction between input/output makes clear that input types can be linked to output types. The high level language can be a natural language interface connected to a Data Base Management System, with data as low level language output. There can be interfaces with more than one dominating link. For instance, the command language input of operating systems is linked to both simple and high level language output. Simple devices are used for manipulating spreadsheets as well as source code. When all possible combinations are put together in a schema (Schema 3) this results in 16 input/output

sequences characterizing a user interface. The input examples in Schema 3 are the same for all output levels and the output examples are the same for the input levels. Not all cells in the table give actual or feasible interface types. The main goal for presenting this space of user interface interactions is to demonstrate the different semantic distances in input and output links.

What has been said about the entries in Schema 1 holds for the entries in Schema 3. They are examples depending on a specific class of users one has in mind. In other words, a choice for entries will, in the context of simulations, strongly depend on characteristics of the learner. Nonetheless, the schema can provide important support in designing actual interfaces. It forces the designer to address in an explicit way the distance and engagement associated with certain classes of learners. This general issue will be a recurring theme when we discuss the concept of 'fidelity level' in the context of generic entities in the learner interface.

4. Generic entities of the learner interface

The four themes ('domain models', 'learning goals', 'learning processes', and 'learner activity') that we distinguished as characteristic of instructional use of simulations (see Section 1 and de Jong, this volume) will be approached in the present paper through what we call the *generic entities* of a user interface for learning with simulations. In our opinion every interface in the context of an ISLE (Intelligent Simulation Learning Environment, see de Jong, this volume) will contain these generic

Entries in space of interfaces	<i>high level language</i>	<i>low level language</i>	<i>low level world</i>	<i>high level world</i>
INPUT	text via keyboard	command language	simple devices	natural manipulations
OUTPUT	printed text	numbers, formulas	simple objects	natural objects

Schema 2 *Examples in a split space of interfaces*

OUTPUT	INPUT			
	<i>high level language</i>	<i>low level language</i>	<i>low level world</i>	<i>high level world</i>
<i>high level language</i>	speech manipulating text	commands manipulating text	mouse manipulating text	Foley's glove manipulating text
<i>low level language</i>	speech manipulating formulas	commands manipulating formulas	mouse manipulating formulas	Foley's glove manipulating formulas
<i>low level world</i>	speech manipulating icons	commands manipulating icons	mouse manipulating icons	Foley's glove manipulating icons
<i>high level world</i>	speech manipulating objects	commands manipulating objects	mouse manipulating objects	Foley's glove manipulating objects

Schema 3 Example entries in Schema 2

entities, which are *the model entity*, *the learning entity*, and *the control entity*. Here we will describe these entities succinctly. In the next sections they will be treated in more detail. Then it will become clear that the four themes can be conveniently and sufficiently covered while exploring the nature of the generic entities. It must be emphasized that the distinction between the entities is a conceptual one. It does not imply that every actual screen that is presented to the learner will always consist of these three entities, nor that the elements that constitute these entities are always grouped together on the screen.

The model entity

This entity of the learner interface contains everything related to the domain model. In line with the distinctions made in the previous section, we will further decompose this entity into an *output* and an *input* part. The output part is used to communicate with the learner about the model. This can be either the model itself, the state of parameters and variables in the model, the 'reality' as reconstructed by the model or critical properties of the model. The input part is used to give the learner the opportunity

to effect changes in the elements (variables and parameters) that determine model states. As we do not deal in this volume with modelling in the sense of allowing the learner to change the model itself, the input from the learner will mostly consist of changing the values of variables and parameters of the model. A special note about a feature which is quite common in many simulations must be made here. The design of measuring equipment by the learner in order to observe states of the model sits somewhere between the classification in input and output. Though a learner input to the computer it serves as an output for the model. To resolve this conflict we will subsume these types of learner actions under the input part of the model entity. If we assume that the model not only includes the model but also associated measuring devices, changing these devices can be seen as modifying aspects of the model.

The distinction between output and input in the model entity almost coincides with two of the four themes. The *domain model* theme is covered by exploring the output part and the *learner activity* theme by exploring the input part.

The learning entity

Everything that is explicitly related to the learning process of the learner is part of this entity. Examples are ways of showing instructional goals, all kinds of learner modifiable support features (scratch pads etc.), directive support such as advice and guidance, navigational support etc. A further decomposition of this entity is possible. In the learning entity there are features which cannot be changed by the learner, they are 'soft wired' by the designer. On the other hand there are elements which are explicitly designed to be modified by the learner. The first will be labelled *instructional* aspects, the second *learning process* aspects. They respectively cover the remaining two themes: learning goals and learning processes.

Two remarks must be made about this entity. First, it is self evident that in designing the input and the output elements of the model entity, the learning processes of the learner will be taken into consideration. The way the model is shown (or not shown) to the learner will greatly influence what and how something is learned. However this 'instructional meaning' of the output (and input) is mostly hidden from the learner. As a contrast, everything in the learning entity is explicitly shown as related to the learning process.

Second, it must be noted that in this entity the learner can and will be active in the sense that s/he provides input to the system. However as 'learner activities' have been defined in Van Berkum et al. (this volume) as actions of learners that impinge on the model, not all actions that take place in the learning entity will be covered by the definition of learner activities.

The control entity

A simulation program is a computer program, just like any other program. Thus it requires a superordinate control entity that gives the user the possibility of influencing the way the program is running. Examples of control actions are quitting, saving, printing, redefining the screen lay out etc. From an instructional point of view this entity is less interesting and that is why we will keep the treatment of this topic rather short. This does not imply that we think this entity to be unimportant. The availability of a learner friendly and congenial

control entity is of the utmost importance in making an instructional simulation acceptable and usable for a wide variety of learners. However we think that an in depth discussion is outside the immediate scope of this paper.

5. The model entity

In this section we will analyze the model entity of the learner interface. This will be done according to the distinction between the output and input elements. Before turning to these elements two concepts which will permeate our discussion of the model entity will be elaborated: model and fidelity level.

5.1. What's a model?

In this article we will take a layered view (Zeigler, 1976) on what constitutes a model in a simulation (see also van Joolingen & de Jong, this volume). Four layers are recognised:

- the system that is modelled (the object system);
- the model of the simulated system outside the machine;
- the representation of that model inside the machine;
- the presentation of the model in the machine to the learner.

For the learner interface the fourth layer is the most important. More specifically the question is how we can represent the model of the object system in the machine to the learner. The representation should allow the learner to have a 'view' (or a number of 'views') of the model, in order to get an understanding of it. The learner interface has to permit a transparent view of the model. At the same time the interface should permit the learner to manipulate the model (learner activity).

Van Joolingen and de Jong (this volume) present an

overview of internal characteristics of simulations. The two main dimensions introduced there are qualitative versus quantitative and continuous versus discrete. It should be noted that this distinction refers to internal (formal) models in the simulation, i.e. the third layer described above. Those internal characteristics alone, however are not sufficient to focus the discussion. In order to become more specific, a domain related classification of simulations is necessary. In terms of the layers: there should also be a link between the first and the fourth layer. This domain related classification should be fairly general in order to avoid too much detail. A classification that satisfies this was given in van Joolingen and de Jong (this volume, Section 2.1.3). This classification is based on a distinction between different kinds of object systems represented by the model in the simulation:

- a) Models representing some *physical process, object or system*. What is meant here is, that the simulation addresses some 'natural world'.
- b) Models representing an *artificial system*, i.e. a man-made object and the states the object can be in (e.g. VLSI, airplanes etc.)
- c) Models representing an *abstract or hypothetical system*. In this last class there is neither a physical counterpart nor an artefact. Examples are models representing abstractions from economics (supply and demand) and mathematical functions not used for modelling either a) or b)

This distinction between different model types will serve as one of the dimensions along which we will discuss the output elements of the model entity.

5.2. Fidelity level

An important aspect of learner interfaces is the fidelity level or verisimilitude (Cunningham, 1984) of the representation on the screen. Fidelity level means the resemblance between the (physical) appearance of the simulation and the 'real world' model it simulates. The fidelity level can refer to both the input and output aspects of the model entity.

There are two broad criteria on which fidelity might be assessed. The first one is how closely the underlying model resembles the 'real world' model. The second one refers to the 'look and feel' of the simulation.

The correspondence between the real world model and the model in the simulation creates the basis for the fidelity level of the simulation. In van Joolingen and de Jong (this volume), we have seen that this correspondence is determined by a so called *experimental frame*, that places a filter over the real world model. All kinds of considerations may play a role in this experimental frame, instructional considerations among them. Making simplifications of reality is generally seen as one of the advantages of computer simulations (Reigeluth & Schwartz, 1989) taking as an assumption that full reality might be too overwhelming for (novice) learners. Several systems provide multiple levels of complexity that can be chosen according to the knowledge level of the learner (e.g. IMTS, Towne & Munro, 1989).

The other criterion is the 'look and feel' of the simulation. Here we can distinguish three dimensions:

- a. Output fidelity, does the output of the system resemble the modelled system closely (in vision and sound)?
- b. Input fidelity, is the way input is provided the same as in reality, both in the kind of data input as well as the way this is done?
- c. Associated to a and b, but worth treating separately, is time fidelity. Does the runtime of the simulation equal the timing as found in reality?

The notion of 'look and feel' is closely related to the concepts of distance and engagement introduced in Section 3.1. These concepts can be seen as a more precise way of defining the rather vague idea of 'look and feel'. This link implies that it will be difficult to make definite statements about the 'optimal' fidelity level in a simulation, because, as we have seen, distance and engagement cannot be established in absolute terms, but depend on the actual class of learners that are the target users of the simulation.

In general there are two considerations for choosing a high fidelity level: *transfer* and *motivation*. Tasks that need to be transferred to reality will profit from a high fidelity level and the level of motivation (sometimes indicated as an inherent characteristic of all kind of simulations) might also be enhanced by a high fidelity level. But, as Reigeluth and Schwartz (1989) indicate, not only overload of information for the learner but also *cost* might be a reason to reduce the fidelity level. Though this may be true in general, in practice the fidelity level must be carefully determined for each specific simulation. That's why fidelity level is the second important dimension used in exploring the model entities.

5.3. The model entity: output

Principally there are two ways of showing the model:

- a. *Covert*, this means that primarily input variables and output states are shown.
- b. *Overt*, parts or properties of the model (or the model as a whole) as they exist in the machine are shown to the learner

The *covert* way of showing the model is essential to learning with computer simulations. It calls upon the exploratory, modelling capacities of learners. Learners have to derive (induce) internal characteristics of the model from the input-output relations. This implies that only input-output relations are shown. This can be done in a number of ways. We can draw upon 'classical' literature on designing texts (e.g. Hartley, 1978) because we could assume there is no fundamental difference between showing a graph on a screen and in a book, except for time relations that you might see emerge on a screen. A discussion of advantages and disadvantages of a number of output representations (e.g. diagrams, bar graphs etc.) and their function is given elsewhere (McKenzie & Padilla, 1984; Mokros & Tinker, 1987). When input-output relations not only refer to specific values of input variables and parameters, but also to ranges of these values, the distinction with overt presentation gets blurred.

Overt presentation refers to showing the model (or parts of it) directly to the learner. The crucial problem is of course how and when to show the model to the learner. The *how* is clearly a learner interface issue and will be discussed in this section.

More generally the following aspects of a model can be shown:

- concepts
- relations
- complex relations (e.g. in the form of diagrams)
- abstract lay-out of the model
- processes
- associated procedures
- critical properties

As a vehicle for showing the model to the learner, use can be made (maybe in adapted form) of modelling techniques as offered to authors (see Van Joolingen, Castells, & Abreu, 1990). Learners might be given the opportunity to zoom in or out on elements of the model. Here we can also present the mathematical background of the model to the learner (maybe in adapted form).

A distinction, which will not be elaborated further, can be made between *main output* and *auxiliary output*. Main output reflects the new state the model is in after the manipulation of input variables. It represents the key aspects of the model. Auxiliary output is not strictly necessary for gauging the state of the model, but can be of great help in determining the next change in the input variables. If applicable, these kinds of output can be placed in different (sub)windows of the general output window. An excellent example of auxiliary output can be found in Flight Simulator where the bearing of the plane is given. This in itself does not give information about the state of the model that is not already available elsewhere on the screen. But this bearing can be of great help in holding the general direction of the plane, especially during landing.

The discussion that follows will explore a two-dimensional matrix whose dimensions are the type of model (physical, artificial, abstract), and the way of presenting the model (overt or covert). For each cell of this matrix we will review important aspects of the interface with a strong emphasis on the fidelity level.

physical, covert

The most simple way of not showing the model, but only a certain state to the learner, is to produce the value of the output variables in a numerical or alphanumeric format. Most of the time, however, this will not be sufficient because understanding of the model will quite often be based on an understanding of relations between different output variables. It is these covariances that matter. This implies that quite frequently the output must be shown in some relational format, leading to graphical output representations. Unfortunately, representations in more than three dimensions are neither easy to produce nor to understand, limiting the possibilities to two dimensional charts most of the time. It's a designers challenge to make choices in this area. Guidelines can be derived from an instructional strategy, indicating that some relations are more important than others and must be shown first. A specific instance of relations would be time dependencies, which are only relevant for dynamic models. An important heuristic in discovering the nature of a model is to trace its outputs over time. It is difficult to imagine how this output trace can be visualized in another way than through insightful graphs. Long lists of numbers are usually not readily interpretable by learners and should be avoided.

In a number of cases a high fidelity level seems unattainable, simply because the variables involved cannot be 'seen' in the traditional sense of the word. High blood pressure itself cannot be visualized, but a dial of a blood pressure gauge can. This leads to the idea of the fidelity level of the equipment involved. An example of a simulation in which there is a very high fidelity level for the equipment is the Lysimeter program described by Peterson, Jungck and Sharpe (1987). What appears on the screen is almost identical to the experimental setup used in practice for measuring the soil moisture relevant for growing plants. Sometimes, efforts

to attain high fidelity levels flounder on shortcomings in the hardware: the machine is for example not fast enough to give a credible representation of some high speed continuous output process.

physical, overt

The first and foremost question is how to show the model. This directly addresses the question of the fidelity level of the model. In some areas something like fidelity level of a model is in itself ambiguous because there is no 'picture' of the object system other than through a model. For example in chemistry, a molecule is not visible and the way to visualize it is at the same time 'the' model. A molecule can be shown by the familiar representation of text books, but also by multi-coloured balls of different size, as is the case in some advanced simulation models available on high speed high resolution graphical workstations. Moreover, the latter representations can also be rotated in order to get a better insight into their spatial structure. It is important to emphasize, that the question of fidelity level, in showing the model, cannot be separated from consideration of other learning material available to the learner. As it is not realistic to assume that all learning will take place through simulations, one should take into account representations that are present elsewhere. The sound strategy seems to be to stay as close as possible to these other materials, because a multitude of representations will only confuse the learner. On the other hand, computer representations will enable certain insights that cannot be readily obtained through other learning material, for example into the spatial structure of a molecule. In such cases a shift in representation is unavoidable, but care should be taken that mappings between representations are available, or even present in the interface.

As soon as there is a 'picture' of the object system the fidelity level will become related to the question of whether high fidelity levels are necessary for understanding the processes involved. Take for example a model that describes the denudation process of a landscape form. This model will describe the relations between variables like type of soil, amount of precipitation, nature of the vegetation, wind direction and wind force, temperature

fluctuations, steepness of slopes etc. This model will be fairly complex and as such not readily open to inspection by a learner. It might be a sensible thing to replace this model, perhaps for some time, with a stylized pictorial representation of the landscape itself, highlighting some of its salient features. Gradually this stylized representation can be replaced by model elements, depending on the state of the learner.

The discussion above makes clear that one should be careful in making absolute statements about the required fidelity level, precisely because shifting fidelity levels during a simulation or a sequence of simulations can be helpful for the learner. As Cunningham (1984) emphasizes: simulations that are too realistic and complex may not permit the identification of the underlying model.

artificial, covert

Although this category is part of the matrix, it is not so easy to think of situations where it makes sense, or even where it is possible at all, to hide the model effectively from the learner when the related object system is an artificial system. More generally this raises again the question of what 'the model' exactly is. In Flight Simulator 'the model' is some complicated formula that, given certain flight variables and environmental parameters, calculates the behaviour of the simulated plane. This model is hidden, there is no way the learner can access it. But to state that during the use of Flight Simulator the learner has to discover this model seems nonsensical. First, because it seems almost impossible to do that: the number of input and output states is indefinite so there can be no sensible strategy for deciphering the underlying model. Second, discovering the model in the sense described above, will not be of great help in learning to fly the simulated machine. On the contrary, it is quite likely that detailed knowledge of the model will hamper the mastering of this capability. This is equivalent to saying that the learning goal has changed: from learning the model to learning certain (psycho)motor skills. Learning of this last type is facilitated by, and depending on, the presence of an adequate model of the behaviour of the artificial system in the machine. Learning to drive a car differs from learning how a car is propelled. In the

first case hiding the model is necessary, in the second hiding the model can be used for 'debugging' notions that learners have about how a car is propelled. If they assume that there is a horse somewhere hidden in the car, then not supplying the car for some time with straw, and observing its performance, can help to falsify that notion³.

In order to be able to elaborate on this problem, the distinction between functional models and reductionist, physical models as introduced by White and Frederiksen (1987a,b; 1990) can be used. A good example of how this distinction is used can be found in IMTS (Towne & Munro, 1989). In this system the author can create a functional model of the device by using a library of Functional Objects. The author can define the behaviour of the objects and the connection to other objects. This functional model is the deep model of the simulation. The learner will be shown the device in either its functional form or in the related physical form or (for single objects) in a combination of both. This means that the learner is given an overt view of the underlying model. However, this view is still superficial, the learner will not see the behaviour of the objects. By attaching measuring instruments (e.g. a multimeter) to the device, the learner may observe the results of changes s/he has introduced (e.g. changing a switch). After updating the device the learner can easily identify those objects that have undergone a change.

The display that is used (functional or graphical), is either chosen by the system or by the learner. The complexity of the display can be changed as well, while the diagrams still exhibit accurate behaviours. This holds for both the functional and the physical presentation.

artificial, overt

This seems to be a fairly straightforward category in which a high fidelity level in model output is generally desirable. This is related to the fact that simulations in this category are quite often directed

³The American Indians called a train an "Iron Horse". Some Indians actually tried to catch such a horse by throwing a lasso around its neck and they died in the attempt. They would have been better off with a train simulation package.

at learning operational knowledge (such as certain motor skills). Obfuscation of models or representation of models does not make sense because the goal is to train the learner in manipulating the actual artificial system. If there are no motor skills involved, high fidelity levels become less important. In some cases high fidelity levels may even hamper the understanding of the artificial system, for example in VLSI models or circuitry. Simplified structural representations will do better in those cases than showing the real thing.

abstract, covert

Good examples of abstract models can be found in economics, because many of the phenomena in this discipline are not visible and some have no real existence. Something like 'the market for good X' cannot be 'visited', it is just an abstraction. However the graph relating supply and demand can be shown, and the slopes of the demand and supply curves in fact represent the model for a particular market. Robinson (1985) surveys a large number of games and simulations in economics. Discovering the nature of the model by systematically varying values of variables and parameters seems especially effective for this kind of model because they cannot easily be shown in a book in a static representation that does not show some key properties of the model (for example elasticity of demand in a supply/demand model from economics).

abstract, overt

Showing an abstract model to the learner who uses a simulation requires careful timing and design. Presenting a complicated mathematical formula, suddenly in its 'pure' form, to the learner, will not be of help in understanding the model. So if the abstract model is shown, this generally should happen at the end of a (number of) session(s). Refinement in the interface should help in linking learning elements to specific elements in the model. It can be shown for example that certain outputs are a result of specific properties of (elements of) the model. In economics again, a model of a market with supply and demand for a good, will lead to specific outputs. If for example the demand does not increase after lowering the price this is related to the elasticity of the demand curve. This in turn is a result of a mathematical model. Showing this

model and indicating at the same time that the observed effect is due to some specific identifiable and, in the interface, highlighted property of the model can be a very effective way of combining output results and showing (parts of) the model.

5.4. *The model entity: input*

The key property of learning with computer simulations is that learners can manipulate the simulation. They can set parameters, change input values etc. The learner interface plays an important role here, in a number of ways.

Van Joolingen and de Jong (this volume; see also van Berkum et al., this volume) distinguish five main groups of learner activity:

- a. defining experimental settings
- b. collecting data
- c. procedure choices
- d. metacontrol over the simulation or setting constraints
- e. data presentations

We will discuss the first four in more detail below because they are clearly part of the input side of the model entity. At the end of this section we will say something about the fifth.

a. *Defining experimental settings*

The learner interface should provide 'physical' handles on the model, i.e. it must enable the learner to change something in the machine. The handles on the simulation enable the learner to manipulate the model and should not frustrate the learner in executing this manipulation.

Two aspects can be distinguished:

- *Choice of variables and parameters that must be changed*
The learner has to decide which of the variables and parameters s/he will change. Sometimes models contain a large number of these variables and parameters (in the case of the medical simulation HUMAN (Coleman & Randall, 1986) the number of variables and parameters

that can be changed is about 200). This implies that somehow the learner should be offered an overview of possibilities. For HUMAN this is accomplished by providing the learner with lists of the variables and parameters. Simulations that use a *graphical interface* and depict the model in terms of related components, mostly have the ability to depict all the components that can possibly be manipulated (if the model is too large for the screen, the learner can scroll through the window). This provides the learner with a graphical overview of possible manipulations.

In non-graphical interfaces the selection is performed by highlighting the entity to be changed and selecting it (mostly by means of pressing ENTER). In graphical interfaces the entity is selected by pointing and clicking the mouse.

- *Changing the values of the components*
Changing the value of a selected parameter or variable can be done in different ways. Basically there are three different ways.
 - a. The learner selects a variable indicating s/he would like its change the value, and is offered a prompt to provide the value. Error messages are given if illegal values are entered.
 - b. The learner is offered a spreadsheet containing the selected parameters and variables and s/he can change their value(s) in the spreadsheet.
 - c. Values of variables and parameters can also be changed by means of direct manipulation (as defined in Section 2.3) of icons like dials and levers⁴.

Just entering a number will be sufficient for most

physical processes, as these do not in reality have dials or levers. Thus the most direct way to enter data is to give the name of the variable and supply a value. This approach is taken for example by HUMAN (Coleman & Randall, 1986), and it seems an acceptable approach given the nature of the physical object system. There is, however, a different situation whenever the physical object system can only be manipulated through experimental equipment. In such situations the alternative is to produce close input equivalents of the real equipment on the screen, giving a high fidelity level of the manipulation equipment instead of the model. This means a shift to a direct manipulation approach. The input device should reflect the nature of the manipulated variable: discrete devices for discrete variables, continuous devices for continuous variables. Confusing them, for example employing a two state switch for an essentially continuous variable, will quite likely bring the learner into difficulties.

Things are different when we deal with abstract processes. In most cases, especially in economics, there is no physical equivalent of manipulation of input variables. This leaves the field free for all kinds of more or less creative solutions. One should beware however of too much creativity; this may confuse the learner. A good example of a simple and straightforward 'no-nonsense' interface as far as input is concerned is Smithtown (Shute, Glaser, & Raghavan, 1989). All input is given through sliders which enable the learner to select input values on the relevant input attributes. No effort is made to create fancy analogies and that seems the appropriate way to do it. For example the population level can be manipulated by moving a slider on a scale between 0 and 100. A more 'creative' way to do this is to represent the population level as a row of human-like symbols one encounters frequently in statistics. By adding such a symbol one could express a rise in population. Implementing this representation however will cost much more and it remains an open question whether adding a human like symbol to a row is more meaningful than manipulating a slider. The example given above does not imply that simplicity is always preferable. This obviously depends also on the characteristics of the learner. Adding a human-like

⁴Direct manipulation may sometimes solicit nonexistent behaviour. An example of this can be found in Baylé (1988). Here users can indicate weights of attributes of objects by enlarging the rope that connects a weight with the attribute. However, in reality the rope gets longer *because* the weight gets larger and not the other way around. A second example is the interface where learners may raise the temperature of an object by raising the indicated temperature on a thermometer that is attached to the object.

symbol can be valuable when the learners are not students but, for example, first level high school pupils. It just serves as a warning that when the field is free, because no natural representations or convincing analogies or metaphors exist, simplicity and consistency can be more important than creativity.

b. *Collecting data*

Though collecting data is strictly speaking part of the output of the model there is a number of simulations that give the learner control over different representations of the output. This kind of control clearly belongs to activities of the learner. The choice for some kind of data reflects the learner's knowledge of the model and the associated procedure. In for example HUMAN (Coleman & Randall, 1986), the learner is able to choose from a large number of variables/parameters that represent the condition of the patient. The choice of the right ones is crucial from the learner's point of view. As we have seen already, in other simulations the learner is free to attach 'measuring devices' (test equipment such as voltmeters etc.) wherever they wish. So s/he not only manipulates the input, but also determines him/herself where and how to obtain the output. In for example ELAB (Böcker et al., 1989), learners may attach measuring instruments to electronic circuits. These measuring instruments include an oscilloscope, a frequency analyzer and analog and digital gauges. Placing these instruments is performed by using the mouse as a pointing and dragging device. Settings of the instruments may be changed using pop-up menus. The instruments have a physical resemblance to real hardware instruments. Böcker et al. (1989) describe ELAB as using a *lab metaphor*, instead of the well known desktop metaphor.

c. *Choosing actions within procedures*

When the simulation is used to teach a procedure or a skill (operational knowledge), not only the value of variables and parameters is important, but also *when* these values are entered. This choice will be influenced by the instructional goal(s) or even the learning goal(s).

Two possibilities exist for this aspect:

- The learner is offered a discrete list of possible actions to perform, and s/he chooses one of them. The list is a sublist of all possible actions available in the program and selected by the author of the program. An example can be found in Protein Purification (Booth, 1987). In this simulation program learners take a series of steps while purifying proteins. In one step they choose (for example) the separation method (e.g. gel filtration or heat treatment). This choice is made by selecting from menus. After performing one step, they may choose what to do next from a new menu⁵. Also certain data (e.g. the record of purification) are only available at specific moments.
- The learner has all possible actions in the program available during the interaction. A good example here is Flight Simulator. The learner has all output and input devices available all the time. The selection has to be made by the learner.

d. *Setting constraints*

Setting constraints on the simulation concerns activities of the learner that control the overall way the simulation is running. Constraints on the simulation are mostly set by an external agent (the tutor for example) but in some cases the learner might choose to explore a simulation under a specific constraint. The interface should help the learner in easily setting and inspecting the constraint. A familiar example is the control over simulation time that the learner might exert. This helps the learner to slow down or speed up a certain simulation process. Some simulations explicitly ask the learner for a time scale before the simulation is run. The predator-prey model described by Svanaes (1990) is an example of a simulation in which setting the pace of the simulation is crucial for understanding the model.

⁵These menus may consist of actions that are quite different, such as asking for data, changing the data representation, control over the setting (go home). Sometimes even steps in the same menu are interdependent (before going to a next step the learner first has to pool fractions, a choice from the same menu).

We also need to consider ways in which the learner can influence the mode of presentation of (model) outputs. Many simulation environments offer the learner data either in a graphical or tabular form. Another, widely used, example of influencing data presentation is switching on or off a grid for graphical output. These kinds of actions by the learner are somewhere in between the output and input aspects of the model entity. They are not learner activities because they don't influence the model as such, neither are they model outputs as defined before. This highlights the unavoidable problems associated with nearly every classification scheme one can choose in this area: none is complete. As we are not able to resolve this problem in any other way than redefining our classification scheme, which will suffer the same fate for another group of learner actions, we just point to this class of learner actions in order to emphasize their importance.

6. The learning entity

6.1. *Instructional aspects*

One of the characteristics of learning with computer simulations is the presence of an instructional goal (see de Jong, this volume). Instructional goals originate with the instructor/teacher and should therefore be distinguished from learning goals which are part of the 'belief system' of learners. The main function of the interface in relation to instructional goals is to communicate these goals to the learner. Support for reaching goals or subgoals is closely related to supporting learning processes and is therefore discussed in the subsection that deals with the learning process aspects.

When defining instructional goals, there is, of course, a strong intertwining between the domain to be taught and the instructional goal. First, the domain representation must be compatible with the instructional goal (see e.g. Gott, 1989). Second, instructional goals are usually stated in a mixture of general and domain specific terms. In the present section we will try to abstract from the domain and restrict ourselves to a pure description of communicating instructional goals, defined in a domain independent way.

Communicating an instructional goal to the learner is making clear what is expected from the learner at the end of (a part of) the instructional setting. It involves the description of domain knowledge (both in content and in structure), of procedures and skills, (including characteristics such as fluency and speed) and/or the description of knowledge acquisition skills (related to their area of application) (see van Berkum & de Jong, this volume). It needs to be stressed that the instructional goal is the target behaviour of the learner, and it is not concerned with 'intermediate' goals the learner will set him-/herself in order to reach the final target behaviour.

There are several ways to communicate the instructional goal to the learner:

- *Presenting the learner with a goal frame*
A goal frame is a screen (window) containing text that describes the knowledge the student has to obtain after interaction with the simulation. There are several books that provide guidelines on how to present instructional goals (Alessi & Trollip, 1985; Hannafin & Peck, 1988). One of these guidelines is to describe the conditions under which the target knowledge has to function. This is especially relevant for the learning of some skills, where characteristics such as accuracy and speed have to be defined. One of the learning goals of simulations can be to learn *knowledge acquisition skills*. This type of skill is quite hard to pin down in a goal frame. In any case the range of applicability of the knowledge acquisition skill should be mentioned in the frame.
- *Showing the target knowledge or skills*
Learning with simulations, takes the learner through a process of exploration, where the target knowledge or skill is rarely explicitly shown in advance. However, this target knowledge or skill (procedure) is represented in the internal representation of the model. This representation can be of different types and at different levels. Hartog (1989) argues that the model as it is represented in the computer should resemble the model that the student is to acquire. This is accomplished in QUEST for example (White & Frederiksen, 1990). Accord-

ing to Hartog *qualitative simulations* form a good basis for making 'teachable' representations⁶. These representations can be shown to the learners as the target knowledge. Showing the target knowledge can also be applied when the learning involves a skill. In Flight Simulator the learner is shown what a perfect landing looks like.

Learning with a computer simulation is a complex task involving complex objectives. Therefore, it is desirable that the learner should be able to inspect the instructional goal during the session, a feature that is hardly ever seen. A notable exception can be found in the system for algebra word problems designed by Singley (1987) and reported in Self (1988). Here learners are presented with a 'goal tree' for solving algebra word problems. Students mark which subproblem they are solving, and the system indicates which subproblems are already solved.

6.2. Learning process aspects

Learning with a computer simulation activates certain learning processes related to exploratory learning. However, no clear comprehensive studies exist about the learning processes that could take place during exploratory learning (see de Jong & Njoo, 1990). At least three functions seem important for support in this area. First, the interface has to support exploratory learning at the levels of *planning* and *navigation*. This means supporting the separation of the study process in a number of phases, with more specific learning processes included and the possibility of inspecting previous actions. Second the interface should support goal attainment. Third, the interface has to support specific learning processes.

6.2.1. Planning and control support

Planning in an exploratory environment is crucial in order to have the learning and interaction process under control. This requires considerable self regulatory capabilities of learners, and supporting these through the interface is highly advisable. However, planning can take place at several levels.

First, there is the level where the study process is subdivided into a number of *stages*. Goodyear et al. (this volume) distinguish four main stages in the learning process: *orientation (analysis)*, *hypothesis generation*, *testing*, and *evaluation*. Dividing the learning process in these stages can be considered as high level planning. A possibility therefore is to guide the learner through these stages. However the learner must be able to switch between stages at will. This means that the interface should facilitate this behaviour. A solution could be to provide three windows that each can be used for a specific stage. For example, in the orientation stage the learner can be offered a goal frame, a goal statement sub-window, and a tool to ask for or investigate (explore) necessary prior knowledge which seems to be a crucial facility in order to make the simulation effective. See e.g. Njoo & de Jong (1991). This example also shows that learners must be allowed to return to a previous stage, for example to update (or inspect) their goal decomposition.

During each of the main stages a number of more specific processes take place that also need some planning, giving another planning level. For example, for each goal a sequence of specific learning processes and learner activity can be outlined. However, there is very little empirical evidence about the contents of interacting with exploratory learning environments. Thus, ideas about how to support planning at this second level can only be highly speculative.

It cannot be denied that there will be a number of different ways to go through the study process and providing support as described above will limit the learner in his/her freedom to explore as s/he likes. These limitations however are only present at a process level, and do not prescribe the behaviour of the learner completely. At the other extreme we can think of planning tools that are completely 'empty'

⁶As Hartog (1989) says: "Apparently the student is not expected to build the same mathematical model that is implemented on the computer when the model consists of, say, 20 or even 200 differential equations" (p. 196).

and just help the learner to schedule the study process⁷. The instructional design process should decide how much 'guidance' should be included in the interface, taking into account learner characteristics and domain characteristics.

Closely related to the aspect of *planning* is the aspect of *navigation*. One of the significant problems with exploratory learning environments is that learners can get lost somewhere in the process. This holds for simulations as well as for hypertext (Hammond, 1989; Jones, 1989). Planning and tracking tools as discussed above might help the learner in navigating through the simulation. If the learner is offered means (e.g. scratch or note pads) to lay down his/her intentions and plans, this might help the learner keep track of his intentions and actions. Alternatively, the system might provide the learner with a simple trace of his/her actions. These traces or scratchpads can also be utilized for helping the learner in *controlling* his/her study process (see also Hardman, 1989).

6.2.2. Support for goal attainment

Frequently, instructional goals are quite comprehensive. Learners have to master a number of (complexly interrelated) concepts, highly complex procedures, and hard to pin down knowledge acquisition skills. Often a need to break down instructional goals into less complex subgoals arises. Depending on the instructional strategy used (which will be selected partly on the basis of learner characteristics), these subgoals will be set by the system or by the learner.

In traditional CAL, instructional goals are normally set by the system (author) and the only choice a learner has, is to choose from a list of instructional goals. In providing support, use will be made of a model of the developing knowledge of the learner in relation to a learning goal. However, in exploratory learning environments, the goals to reach are

often set by the learner him/herself. Thus it will not always be obvious to the system which subgoal the learner is pursuing. The system needs to infer these subgoals from the learner's behaviour in order to store this information in the learner model. By supplying the learner with a tool to state and update the subgoals, an input for the system to infer these subgoals is created (see Twidale, 1989).

Supporting the learner in decomposing a goal means that the learner must be able to:

- a. select a part of the target knowledge (the underlying model) and to work with it independently. In fact this means breaking up the simulation in sub-simulations (this only holds for complex simulations of course) or,
- b. select and exercise subprocedures or subskills separately. These subprocedures and subskills must be combined later, either by placing them in a sequence and/or by exercising a procedure until it becomes 'automated' (compiled), or,
- c. unravel a complicated knowledge acquisition skill into subskills, and practice them separately.

An example of help the learner can be offered in breaking down an (instructional) goal into subgoals, can be found in IMTS (Intelligent Maintenance Training System). In fact this is an authoring system that can be used to create simulations of complex devices (such as a helicopter blade fold system) by using, adapting and creating generic objects and their relations (see Towne, Munro, Pizzini, Surmon, & Wogulis (1988) and Towne & Munro (1989)). Models created with IMTS are used to train learners in diagnosing faults. The top level instructional goal of a model created with IMTS is to teach learners to identify failures in complex devices containing all the objects (variables and parameters) and their interrelations. The system explicitly supports the subdivision of models into submodels, so called *scenes*, containing only a part of the device. This subdivision is shown through a hierarchical map of functional subsystems. Learners can choose a subsystem to practice with, or they can even create subsystems themselves by selecting relevant objects.

⁷An analogy to planning tools as used in management such as PMW, Super Project Expert & MS-project (Janes, 1989) can be found here.

An example of goal decomposition related to skills or procedures can be found in Flight Simulator. The general goal of learning to fly the airplane is decomposed into subskills like taking off, straight and level flying, taxiing etc. These subskills can be practised separately. However there is no specific support for keeping track of the goals that have already been attained. Sequencing the goals is entirely left to the learner, though a preferred sequence is indicated in an accompanying manual.

The instructional strategy used might prescribe a certain task decomposition and thus take away the setting of subgoals from the learner. However, when the setting of subgoals is performed by the learner the interface may offer means to support this process. This can be done by offering tools for it or even by making suggestions about the subgoals. Precisely how the learner will be helped in expressing his/her goals to the system, is currently a subject of research. Twidale (1989) developed a system (EPIC) on propositional calculus. In EPIC learners can select plans from menus and while working indicate in the plan the subgoals that have been achieved, and the subgoals the learner is working on. An empirical study showed that EPIC not only provides information on the goals of students to be incorporated in a learner model, but that the possibility to state and indicate subgoals also helps the learners to explore the domain by reducing the load on working memory (Twidale, 1989, p. 305).

6.2.3. Support for specific learning processes

A list of specific learning processes applicable to learning with computer simulations is given by Goodyear et al. (this volume). Here, we will present some ideas on how the interface might sustain the learner in performing some of those learning processes. The support that we list here is of what we have called a *non-directive* nature. It doesn't direct the learner what to do, but helps him/her to execute his/her intentions in a better, more efficient way. *Directive support* that tells the learner more directly what to do, is discussed in Van Berkum and De Jong (this volume).

Model exploration

Model exploration is one of the learning processes primarily performed in the 'orientation phase'. Exploring the model is related to the overt presentation of a model as discussed in Section 5.3. More generally we can identify three ways in which the interface can support this specific learning process:

a. *Allowing a hypertext like exploration of a model.*

This provides the learner with a direct (expository) way of exploring a model. The learner might be offered a schematic overview of the model and be allowed to explore all kinds of definitions and relations by travelling through a hyperspace (see e.g. Hammond, 1989).

b. *Allowing a change of views.*

For the same domain different views may be available. These views can include simply changing the presentation mode of the model, thus offering adaptability to learner characteristics (e.g. verbalizers/visualizers), but may also involve a genuinely different perspective on the domain. An example of the latter can be found in Coinland/Numberland (Hamburger & Lodgher, 1989). Coinland is an instructional system meant to teach principles in arithmetic, such as multi-digit subtraction. In Coinland operations like subtraction are carried out by having the learner act as a shop-owner who sells an item to a customer. The learner returns money to the customer by manipulating different kinds of coins on the screen. In Numberland coins are replaced by numbers. The learner is able to change from Coinland to Numberland and vice versa at will.

c. *Allowing zooming in/out on domain concepts.*

Some simulations offer the learner the opportunity to get a more detailed view of the model by allowing zooming in on domain concepts. Examples are simulations created with a simulation package such as CHEMCAD (Lipowicz, 1987). Here learners can inspect properties of objects in chemical installations or properties of chemical substances.

The process of model exploration also involves the

learner creating his/her own mental model of the model offered. This process can be aided by giving the learner an opportunity to write down his/her ideas about the model. Sometimes, tools that were used to create the domain models (see van Joolingen, Castells & Abreu, 1989), especially when they have a graphical interface, might be offered to the learner in order to 'write down' his ideas.

Hypotheses generation/predicting

One of the specific learning processes one likes to encourage in exploratory learning is creation and testing of hypotheses. How to stimulate the generation of hypotheses is handled by the instructional strategy (e.g. by providing suggestions). The learner interface might help the learner in expressing the hypotheses. A (dedicated) scratchpad or a spreadsheet might give this opportunity.

An example of this can be found in CIRCSIM-TUTOR (Kim, Evens, Michael, & Rovick, 1989), an ITS in the domain of medicine which treats problems associated with blood pressure. The instructional strategy in CIRCSIM is quite directive. Learners are posed with a perturbation of the cardio-vascular system; for example, 'the atrial resistance is decreased to 50% of normal'. Subsequently they are asked to predict what will happen to seven components of the cardio-vascular system. This prediction is made in a qualitative way (increase, decrease, steady) for three different moments in time. To be able to write down this prediction learners are offered a 7 (components) x 3 (moment in time) spreadsheet. The spreadsheet itself gives the components and the moments in time and is thus rather restrictive. A slightly different approach is taken in PPT (Pathophysiology Tutor, Michael, Haque, Rovick, & Evens, 1989). Here learners can indicate a predefined hypothesis by using a list of nested menus that each give a more specific fixed list of hypotheses in the field of physiopathology. This way of generating a prediction is even more restricted than the way used in CIRCSIM-TUTOR. In Smittown, a microworld for elementary microeconomics (Shute, Glaser, & Resnick, 1986; Shute, Glaser, & Raghavan, 1989), three menus are presented to the learner. The first one contains 'connectors' (if, then, as, when, and, the), the

second one 'variables' (such as: price, population, income, interest rate), and the third one contains 'descriptors' (such as: increases, decreases, equals, is part of). By combining connectors with variables and descriptors, the learner can state his hypothesis (e.g. as interest rate increases, price increases).

Hypotheses may differ on a number of characteristics like *generality of scope*, and *precision of prediction* (Spada, Stumpf, & Opwis, 1989). The learner interface might support the learner in stating his/her hypotheses at different levels of generality and precision.

Important here are *temporary changes*. A learner (especially those not familiar with the domain) would like to test a few ideas quickly and then return to his/her original settings. Moreover, a general possibility to save and go back to intermediate states can encourage students to test hypotheses quickly.

As a concluding note to this section we would like to emphasize that, in developing strongly learner driven systems, designers must be aware of their inherent dangers. They tend to have a large number of different 'commands' for dealing with all the features of the system. As a result, a shift in the way the simulation is approached can occur. What should be a learning exercise becomes an effort to understand or administer a complex system (Cunningham, 1984).

7. The control entity

The control entity in the learner interface deals with everything that is related to the high level control of the environment in which the simulation runs. As such it has much in common with the control entity that is present in almost every other application. As a consequence much that is valid for other applications will also be valid for simulations. It is not our goal to review the substantial amount of literature in this area. Below we will focus on some simulation specific aspects of the control entity.

Saving instances

Though not entirely unique for simulations, the

occasional need to save several instances of model states that occurred during the run poses a specific problem. In order to save meaningful instances, the learner must have the ability either to select or to mark those instances that must be saved for subsequent runs. The terms 'select' and 'mark' indicate two different approaches to this problem. In the case of 'marking' the learner can indicate, for every state of the model that is encountered during the run, whether it should be saved or not. By 'selecting' we mean that the learner can 'visit' earlier states and select those for saving that seem worth it in retrospect. A combination of these two strategies is also possible.

Sequencing

Being an instructional system, a simulation will quite likely be part of an ordered sequence of 'lessons'. As a result learners will occasionally experience the need for a rehearsal of previous lessons. For example one can imagine that during learning to navigate with Flight Simulator, one will once again try to practice straight and level flying. In Flight Simulator you must then quit the 'Navigation Lesson', enter the 'Straight and Level Lesson', quit that one and enter again the navigation lesson. Meanwhile you have lost your original navigation position. Of course this is pretty cumbersome. A better way to deal with this sequencing problem is to 'freeze' the state of the navigation lesson while the learner is in the straight and level lesson. Upon returning to the navigation lesson the learner finds everything as it was when s/he left. Facilitating this sequencing is an important aspect of interfaces for certain types of simulations.

Mode control

There are a number of simulations, especially in the area of motor skills, in which the learner can run the system under complete instructor/tutor/teacher control. The system simply shows the learner how to do it. The other state of the simulation is learner control. Shifting between these two modes is an important aspect of learning with simulation. Though the exact nature of the two modes is in the model and the learning entity, the possibility to change smoothly between the two modes belongs to the control entity. In Flight Simulator the learner has only limited control in this area. As soon as a

lesson is activated the learner can only sit it out or quit and in both cases the system automatically shifts to the learner control state, independent of the needs of the learner. This also has to do with sequencing, because it forces the learner into a sequence that imposes some additional effort to run two lessons one immediately after another.

8. Designing the learner interface

In this section we turn to the important question of designing the learner interface. We will try to present a conceptually consistent ordering of objects and actions (see e.g. the CLG of Moran (1981) which can serve as a starting point for functional design of learner interfaces. The emphasis is on 'functional' because we will not address the question of how to design actual screens.

We see two basic ways for functionally grouping input and output:

- 1) according to the type of action involved
- 2) according to the object(s) of action(s)

The first categorization implies that actions are organized according to a number of generic action classes. Let's clarify this with some examples. A generic action class consists of all actions that are related to help. Another example are all actions related to the lay-out of the screen, like resizing windows. What precisely constitutes a 'generic action' is debatable. One could argue that actions like 'edit', 'modify', 'move' etc. are the most generic ones. But in our opinion it is very unlikely that a learner will plan his/her actions using the generic types. For example the general idea of 'want to modify' seems pretty nonsensical as a learner plan. It seems far more plausible that actions are related to generic classes that have richer semantics than actions like the ones mentioned above. Thus, in our view, generic action classes are related to semantically rich concepts whereas action types like 'modify' are abstract actions that can be part of different pop-up menus related to actions.

The second categorization groups actions according to the type of object they are related to.

example of this categorization in simulations is a class called 'model(s)'. All permissible actions concerning the model(s) are grouped under this header, for example zooming in/zooming out, partial inspection, etc.

In both cases the basic problem is to find what one could call 'generic' classes, that is, classes that, in a way meaningful to the learner, group related actions or objects together. We will now introduce a tentative classification of generic action and object classes relevant for learning with simulations.

Generic action classes for simulation interfaces

Below a (preliminary) list of generic action classes for simulations is presented:

- *lay-out actions*: all actions related to rearranging the lay out of the screen (for example setting colours, resizing windows, dragging windows, iconizing windows, setting data representations)
- *help actions*: all actions connected to obtaining help from the system. This help is at the keystroke level, not the instructional level.
- *learner control actions*: all actions related to running the simulation environment as a whole (examples: quitting, starting, saving instances, go to next lesson etc.).
- *learner activity actions*: all actions related to changes the learner wants to effect in the simulation itself. This class subsumes actions like defining experimental settings and supplying data.
- *learning goal actions*: all actions related to the learning goal aspects of the simulation. This class includes actions like activating a goal frame, invoking examples of target skills or knowledge etc.
- *learner process actions*: this class subsumes actions like planning the session, stating and verifying hypotheses, inspection actions (model exploration, change of views), goal decomposition and navigation, asking for guidance (explanation and feedback).

As can be seen, most of these generic action types quite naturally tie in with the main entities of the learner interface discussed in this paper.

Generic object classes for simulation interfaces

Below we will list a number of generic object classes that can play a role in simulations:

- *model objects*: the model or all models that are part of the simulation software. Actions possible on this object comprise all permissible (passive) actions that can be applied to the object, for example zooming in/out, activating visualizers, showing partial representations etc. This class will only be present when the simulation is overt.
- *model variable/parameter objects*: the constituting elements of the model *without* their connectors in the model. The most frequent action that will be performed on these objects will be changing their values, but one can imagine simulations in which a learner can obtain more information about these objects, like measurement level, critical values etc.
- *hypotheses objects*: the hypotheses that will be tested during the simulation. Actions are for example stating the hypothesis, viewing the hypothesis, exploring relations between hypotheses, refining hypotheses, checking the state of a hypothesis. This class also comprises predictions about the expected state of the system after some data input.
- *goal objects*: objects that represent the learning goals for the simulation session(s). These can be arranged in higher order structures like trees, which can be built, changed and explored. Associated actions could be inspecting the goals, modifying the goals, updating the goals etc.
- *plan objects*: objects that are part of a plan for the simulation session, i.e. incorporating some sequence or path in time.
- *system output objects*: objects that are related to messages of the system (for example prompts etc). Actions related to this object are control

actions on the simulation like quitting, saving etc.

- *directive support objects*: objects that contain information about advice and guidance related to the simulation's content. Actions are consulting an online manual, obtaining external non-dynamic guidance etc.
- *screen objects*: objects that are part of the screen (windows etc). The actions are well known things like dragging, resizing etc.

The basic question is of course which categorization should be used for designing the learner interface. From the viewpoint of consistency, it seems preferable to select one categorization and stick to that as much as possible. But in practice things turn out to be different, because in some cases it is not possible to find an object-like equivalent. If one decides for an object categorization, some indispensable actions cannot be readily attached to meaningful objects, for example help actions. If one, on the other hand, chooses an action categorization, there are a number of actions that are so closely related to the objects they operate on that they cannot be considered any longer as belonging to a generic class. For example, expressing and changing a plan for a simulation session will require actions that are more or less unique for this object. This implies that in practice some mixture of the two approaches will be necessary. It should be noted that most existing interfaces also contain this mixture. The MacIntosh top menu bar, for example, contains entries which refer to actions (Shut Down) and entries which refer to objects (Calculator).

To conclude this section, we want to stress that one has to keep in mind the fact that in many simulations input and output is mixed in one window. Flight Simulator is again the perfect example of this mix. In this simulation the mix is a consequence of the choice of a high fidelity level. This holds especially for the instrument panel that is a copy of the interior of the average cockpit. One can doubt however whether the lay out of the instruments in the cockpit is suited for learning purposes. The whole panel could be redesigned according to some of the principles laid out above and that would

surely lead to more functional coherence. But the price one has to pay for this is a reduced fidelity level. A halfway solution could be to give output and input categories that are functionally related the same colour while retaining their physical location.

9. Interface demands and hardware facilities

Discussing hardware in the context of educational software is always somewhat tricky. First because there is no glimpse of standardization in the equipment available to the different educational institutions. In reality they range from almost pre-historic 64K machines to advanced workstations. Second, buying computers for educational institutions is more often a question of either national pride (buy the machines your country builds) or very limited budgets, than of supplying learners with up to date equipment that fully operationalizes the power inherent in some software (simulation) packages. Third, technology moves fast. What seems to be good advice today, looks stupid after the next drop in hardware prices. The question addressed in this section is what kind of additional requirements can be put forward concerning hardware, from the angle of simulation.

Of course there is no general answer to this question. Some types of simulations will run perfectly on rather simple machines. For example HUMAN (Coleman & Randall, 1986), a complex simulation in the field of medicine, has no other demands than a 8086 MS-DOS machine, without any graphical output devices or special input devices. Just a keyboard and a screen are sufficient. On the other hand, the EXTEND (TMImagine That Inc.) toolkit relies heavily on the MacIntosh interface, and as a result the simulations built with it will do the same. The easiest way out is to request simply the full panoply of interface functions available on the more advanced machines, because it is always possible to use less than is available. But this strategy has its price, and budgets are often very low in educational institutions. Thus some more sensible advice is needed for the designer of a simulation package/program.

We will focus the discussion around the following

aspects of the hardware: speed, screen size and resolution, colour, input devices, output devices, external media.

Speed

Speed is important in very dynamic simulations requiring high output fidelity. The updating of a screen must happen quickly in those settings, because delays not only slow down the learning process, but also lower the fidelity level. A quick look at Flight Simulator running on a 6 Mhz machine or a 16 Mhz machine will convince everyone. It should be emphasized that this not only holds for motor skill learning, but can also be important for cognitive, higher order skills. Showing rotating molecules in order to understand their structure is neither very dynamic nor does it require high fidelity, but it still takes a supercomputer to perform this work in a reasonable time.

Speed can also be important when different views are offered of a model. It is very demotivating when changing between views takes a lot of time. In this way learners will be discouraged from exploring the model. As for general speed of calculation, we think that the bottom-line machines of today, let alone those of the future, will be fast enough to perform fairly complex calculations on numerical models.

Screen size and resolution

Though the large 19 inch screens are still expensive, there can be no doubt that in the foreseeable future they will dominate the market. The required screen size is of course first and foremost a function of the number of windows, if any at all, the simulation will need simultaneously. Though too many windows will confuse the learner, in some cases use of a number of windows cannot be avoided. Furthermore one should take into account the level of detail that must be observable for the learner. If some crucial component of an artificial system cannot be shown at a satisfactory size, a larger screen is needed. As far as resolution is concerned, in simulations related to some physical process or artificial system, requiring high fidelity levels in input and output and/or overt model presentation, one should strive for good resolution levels. Given the technological advances in this

area, the EGA adapter already furnishes an acceptable resolution and the MacIntosh even a pretty good one. It seems not too far-fetched to expect the general availability of high resolution screens for reasonable prices in the near future.

Colour

There has been and still is a lot of debate about the proper use of colour. In simulations colour can be used for different purposes. The most obvious one is making the simulation as realistic as possible, especially when well known physical or artificial systems are represented. This does not imply that there must always be a very close resemblance between real world colours and colours used in simulation interfaces. In Flight Simulator the countryside is yellow in some instances and the water blue. This is not too disturbing, but the other way round would be quite annoying for learners and lead to very serious orientation problems. The other use of colour is in grouping menus or commands. In simulations one could give all input data messages the same colour etc. This use of colour, even if one employs grouped pull down menus, can help the learner in discovering and remembering the structure of the interface. Another type of use in this category is to use colour to indicate different levels in hierarchically organized menus. This technique is employed by Flight Simulator where all first level options are white on light blue, all second level option yellow on green and all third levels options white or red on black.

Input devices

Input devices range from the archetype keyboard to advanced speech recognition systems and the physical manipulation devices described in Section 2. It is hard to say which device is best suited for which type of simulation and which type of learner. If one requires high input fidelity in domains where equipment is used very often, the natural way is to go for the more advanced devices like mice etc. But in other cases where straightforward numerical data input predominates, a keyboard will be the best choice. One should be aware of those cases in which the non-availability of a proper device makes learning more difficult than necessary. Using Flight Simulator with a joystick or with cursor keys makes an enormous difference in the time it takes to learn

flying the plane. As there is no use in learning to manipulate the cursor keys in actually flying a plane, this situation should be avoided. For strongly menu driven systems the touch screen or the light pen, if affordable, seems to be the natural choice. This reduces user movements to a minimum and ideally one could even do without a keyboard, which after all needs some space on a desk in an instruction room.

Output devices

As far as output is concerned, most things have already been said under the aspects of speed, screen size and resolution. In some cases non-visual output can be important. Though the time that the computer can address our other senses (smell, tactile, taste) still seems remote, hearing can be too easily overlooked in designing the output of simulations. For example in traffic people rely very much on hearing for determining the distance to other vehicles as well as the type of vehicle. A simulation that does not incorporate traffic noise in a credible way, may fall short of teaching the driver the most efficient behaviour. This can even lead to erroneous behaviour in those situations in which you cannot see the other traffic but can hear it, for example in complicated street crossings. In other cases the noise of a machine is indicative of its behaviour, for example the engine of a car. Sometimes the noise is even needed in order to be able to diagnose a malfunctioning machine.

External media

The assumption underlying the discussion of the previous aspects is that most input and output goes through the computer screen. This is of course a gross simplification. Modern technology offers many possibilities for carrying out simulations without even using a computer screen, though a computer program controls what is on the external media. Interactive video is the most advanced example at the moment. However, a videodisc will always contain a number of discrete 'events', that can be addressed by the program. Events are not created at run time, just addressed. Of course, the number of events can be enormous, but in principle the number is limited. This implies that for simulations, where the learner sometimes might want to change a number of parameters, possibilities are

limited. On the other hand, interactive video, provides great opportunities, whenever high (output) fidelity is necessary, and the number of variations is limited.

10. Summary and conclusions

In this article we reviewed literature that is either directly or indirectly related to learner interfaces for simulations. Moreover, we added some ideas of our own about interfaces that close the still considerable gap between the learner and the machine. However, what is still lacking is a coherent and usable theory about how to design effective interfaces in an instructional context. By design we literally mean design: how to produce computer screens that support the learning processes in an effective way. It seems that most of the literature follows what one could label the *screen dump approach*. This approach is characterized by including a considerable amount of examples of actual screens in the report or paper. There is nothing against doing that, but what is patently lacking is an elaboration of the rationale behind the screen: why does it look like it does? Without giving these rationales it becomes impossible to accumulate valuable general design principles for learner interfaces. The author who creates instructional simulations will not only need an advanced User Interface Management System (UIMS) for supporting this work, but probably even more those general design principles without which the UIMS is only a worthless empty tool. There is no substitute for good and proven ideas in this area. Thus the main research challenge was and still is developing a coherent and usable theory for designing learner interfaces for simulations.

References

- Alessi S.M., & Trollip, S.R. (1985). *Computer based instruction, methods and development*. Englewood Cliffs, NY: Prentice-Hall.
- Baylé, H. (1988). *MacMidas*. Report Vakgroep SWI, University of Amsterdam.
- Böcker, H.D., Herczeg, J., & Herczeg, M. (1989). ELAB - An electronics laboratory. In D. Bierman, J. Breuker & J. Sandberg (Eds.), *Pro-*

- ceedings of the 4th international conference on AI & Education* (pp. 15-25), Amsterdam: IOS.
- Booth, A.G., & Hames, B.D. (1987). *Protein Purification*, Oxford: IRL Press.
- Coleman, T.G., & Randall, J.E. (1986). *HUMAN-PC: A comprehensive physiological model [Computer program]*. Jackson: University of Mississippi Medical Center.
- Cunningham, J.B. (1984). Assumptions underlying the use of different types of simulations. *Simulations & Games*, 15, 213-234.
- Fiderio, J. (1988). A grand vision. *Byte*, 13(10), 237-244.
- Foley, J.D. (1987). Interfaces for advanced computing [Special issue; Trends in Computing]. *Scientific American*, 1, 62-67.
- Foley J., & van Dam, A. (1982). *Fundamentals of Interactive Computer Graphics*. Reading (Mass.): Addison-Wesley.
- Gaines, B.R., & Shaw, M.L.G. (1986a). From timesharing to sixth generation: the development of human-computer interaction: 1. *International Journal of Man Machine Studies*, 24(1), 1-27.
- Gaines, B.R., & Shaw, M.L.G. (1986b). From timesharing to sixth generation: the development of human-computer interaction: 2. *International Journal of Man Machine Studies*, 24(2), 101-123.
- Gasper, E. (1988). Getting a head with Hyper-Animation. *Dr.Dobbs Journal of Software Tools*, 13(7), 18-34.
- Gott, S.P. (1989). Apprenticeship instruction for real-world tasks: The coordination of procedures, mental models and strategies. *Review of Research in Education*, 15, 97-169.
- Hamburger, H., & Lodgher, A. (1989). Semantically constrained exploration and heuristic guidance. *Machine-Mediated Learning*, 3, 81-107.
- Hammond, N. (1989). Hypermedia and learning: who guides whom? In H. Maurer (Ed.), *Computer Assisted Learning. Proceedings of the 2nd International Conference ICCAL* (pp. 167-182). Berlin: Springer Verlag.
- Hannafin, M.J., & Peck, K.L. (1988). *The design, development, and evaluation of instructional software*. New York: Macmillan.
- Hardman, L. (1989). *Requirements of navigation metaphors for a hypermedia learning environment*. Office Workstations limited. SAFE/HYP/OWL-pap.
- Hartley, J. (1978). *Designing instructional text*. London: Kogan Page.
- Hartog, R. (1989). Qualitative simulations and knowledge representation for intelligent tutoring. In H. Maurer (Ed.), *Computer Assisted Learning. Proceedings of the 2nd International Conference ICCAL* (pp. 193-213). Berlin: Springer Verlag.
- Hill, R.D. (1987). *Supporting concurrency, communication and synchronisation in human-computer interaction* (Technical Report CRSI-197). Master's thesis. Computer Systems Research Institute, University of Toronto, Canada.
- Hudson, S.E. (1987). UIMS support for direct manipulation, *Computer Graphics*, 21(2), 120-124.
- Hutchins, E.L., Hollan, J.D., & Norman, D.A. (1986). Direct manipulation interfaces. In D.A. Norman & S.W. Draper (Eds). *User centered System Design*. Hillsdale, NJ: Lawrence Erlbaum.
- Janes, G. (1989). More projection, please. *PC World Focus*, 5, 11-15.
- Jones, T. (1989). Incidental learning during information retrieval: A hypertext experiment. In H. Maurer (Ed.), *Computer Assisted Learning. Proceedings of the 2nd International Conference ICCAL* (pp. 235-255).
- Jong de, T., & Njoo, M. (1990). *Learning and instruction with computer simulations: learning processes involved*. Paper presented at the NATO workshop "Computer-based learning environments and problem solving", Leuven (Belgium), September 26-28.
- Joolingen van, W.R., Castells, J., & Abreu, J.L. (1990). Review of simulation construction tools. In: K. Tait (Ed.), *Towards the specification of support tools for authors constructing simulation-based intelligent learning environments*. University of Leeds, Computer Based Learning Unit.
- Kim, N., Evens, M., Micheal, J.A., & Rovick, A.A. (1989). CIRCSIM-TUTOR: An intelligent tutoring system for circulatory physiology. In H. Maurer (Ed.), *Computer Assisted Learning. Proceedings of the 2nd International Conference ICCAL* (pp. 254-267). Berlin: Springer Verlag.
- Lipowicz, M. (1987). Distillation-column design software. *Chemical Engineering*, 94, 167-168.
- McKenzie, D.L., & Padilla, M.J. (1984, April). *Effects of laboratory activities and written simulations on the acquisition of graphing skills by eight*

- grade students. Paper presented at the meeting of the National Association for Research in Science Teaching, New Orleans, LA.
- Michael, J.A., Haque, M.M., Rovick, A.A., & Evens, M. (1989). The pathophysiology tutor: a first step towards a smart tutor. In H. Maurer (Ed.), *Computer Assisted Learning. Proceedings of the 2nd International Conference ICCAL* (pp. 390-400). Berlin: Springer Verlag.
- Mokros, J.R., & Tinker, R.F. (1987). The impact of microcomputer based labs on children's ability to interpret graphs. *Journal of Research in Science Teaching*, 24(4), 369-383.
- Moran, T. (1981). The command grammar language: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, 15, 3-50.
- Njoo, M., & de Jong, T. (1991). *Learning processes of students working with a computer simulation in mechanical engineering*. To appear in the proceedings of the 1989 EARLI conference, Madrid.
- Peterson, N.S., Jungck, J.R., Sharpe, D.M., & Finzer, W.F. (1987). A design to science. Simulated laboratories: Learning via the construction of meaning. *Machine-mediated Learning*, 2(1), 111-127.
- Reigeluth, C.M., & Schwartz, E. (1989). An instructional theory for the design of computer-based simulations. *Journal of Computer-Based Instruction*, 16(1), 1-10.
- Robinson, J.N. (1985). Games and simulations in management and economics education: A survey. *Economics*, 163-171.
- Self, J. (1988). Bypassing the intractable problem of student modelling. *Proceedings of ITS*, Montreal.
- Self, J. (1989). *The case for formalising student models (and Intelligent Tutoring Systems generally)*. Paper presented at the AI&Ed conference, Amsterdam may 1989.
- Shneiderman, B. (1987). *Designing the user interface, strategies for effective human-computer interaction*. Addison-Wesley.
- Shute, V. J., Glaser, R., & Raghavan, K. (1989). Inference and discovery in an exploratory laboratory. In P.L. Ackermann, R.J. Sternberg, & R. Glaser (Eds.) *Learning and Individual Differences* (pp. 279-326). New York: W.H. Freeman.
- Shute, V., Glaser, R., & Resnick, P. (1986). *Discovering and learning to discover: An intelligent microworld for economics*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.
- Singley, M.K. (1987). The effect of goal posting on operator selection. *Proceedings of the third international conference on artificial intelligence and education*. Pittsburgh.
- Spada, H. Stumpf, M. & Opwis, K. (1989). The constructive process of knowledge acquisition: student modelling. In Maurer, H. (Ed.), *Computer Assisted Learning* (pp. 486-500). Proceedings of the 2nd International Conference ICCAL '89 (Dallas). Berlin: Springer Verlag.
- Svanaes, D. (1990). Simulation models + User interfaces = Interactive applications. *Computers and Education*, 14, 363-370.
- Towne, D.M., & Munro, A. (1989). Artificial intelligence in training diagnostic skills. In D. Bierman, J. Breuker & J. Sandberg (Eds.), *Artificial intelligence and education; synthesis and reflection. Proceedings of the 4th International Conference on AI and Education* (pp. 291-297). Amsterdam: IOS.
- Towne, D.M., Munro, A., Pizzini, Q.A., Surmon, D.S., & Wogulis, J. (1988). *Intelligent maintenance training technology* (Technical report No. 110). Behavioral Technology Laboratories- USC.
- Twidale, M. (1989). Intermediate representations for student error diagnosis and support. In: D. Bierman, J. Breuker & J. Sandberg (Eds.). *Artificial intelligence and education; synthesis and reflection*. Proceedings of the 4th International Conference on AI and Education, May 1989, Amsterdam, Netherlands (pp. 298-306). Amsterdam: IOS
- White B.Y., & Frederiksen, J.R. (1987a). Qualitative models and intelligent learning environments. In W. Lawler & M. Yazdani (Eds.), *Artificial intelligence and education, Vol. 1*, (pp. 281-305). Norwood, NJ: Ablex.
- White, B.Y., & Frederiksen, J.R. (1987b). *Causal Model Progressions as a Foundation for Intelligent Learning Environments* (Report No. 6686). Cambridge, MA: BBN Laboratories.
- White, B.Y. & Frederiksen, J.R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial Intelligence*, 42,

99-157.

- Woolf, B. (1988). Representing complex knowledge in an intelligent machine tutor. In J. Self (Ed.), *Artificial intelligence and human learning*. London: Chapman and Hall.
- Zeigler, B.P. (1976). *Theory of Modelling and Simulation*. New York: John Wiley and Sons.