Instructional environments for simulations

Jos J.A. van Berkum
Courseware Europe BV, Knowledge Engineering

Ton de Jong
Eindhoven University of Technology, Department of Philosophy and Social Sciences

The use of computer simulations in education and training can have substantial advantages over other approaches. In comparison with alternatives such as textbooks, lectures, and tutorial courseware, a simulation-based approach offers the opportunity to learn in a relatively realistic problem-solving context, to practise task performance without stress, to systematically explore both realistic and hypothetical situations, to change the time-scale of events, and to interact with simplified versions of the process or system being simulated.

However, learners are often unable to cope with the freedom offered by, and the complexity of, a simulation. As a result many of them resort to an unsystematic, unproductive mode of exploration. There is evidence that simulation-based learning can be improved if the learner is supported while working with the simulation. Constructing such an instructional environment around simulations seems to run counter to the freedom the learner is allowed to in 'stand alone' simulations. The present article explores instructional measures that allow for an optimal freedom for the learner.

An extensive discussion of learning goals brings two main types of learning goals to the fore: conceptual knowledge and operational knowledge. A third type of learning goal refers to the knowledge acquisition (exploratory learning) process.

Cognitive theory has implications for the design of instructional environments around simulations. Most of these implications are quite general, but they can also be related to the three types of learning goals. For conceptual knowledge the sequence and choice of models and problems is important, as is providing the learner with explanations and minimization of error. For operational knowledge cognitive theory recommends learning to take place in a problem solving context, the explicit tracing of the behaviour of the learner, providing immediate feedback and minimization of working memory load. For knowledge acquisition goals, it is recommended that the tutor takes the role of a model and coach, and that learning takes place together with a companion.

A second source of inspiration for designing instructional environments can be found in Instructional Design Theories. Reviewing these shows that interacting with a simulation can be a part of a more comprehensive instructional strategy, in which for example also prerequisite knowledge is taught. Moreover, information present in a simulation can also be represented in a more structural or static way and these two forms of presentation can be alternated. A third conclusion is that learners can be provoked to perform specific learning processes and learner activities by tutor controlled variations in the simulation, and by tutor initiated prodding techniques. And finally, instructional design theories showed that complex models and procedures can be taught by starting with central and simple elements of these models and procedures and subsequently presenting more complex models and procedures.

Most of the recent simulation-based intelligent tutoring systems involve troubleshooting of complex technical systems. Learners are supposed to acquire knowledge of particular system principles, of troubleshooting procedures, or of both. Commonly encountered instructional features include (a) the sequencing of increasingly complex problems to be solved, (b) the availability of a range of help information on request, (c) the presence of an expert troubleshooting module which can step in to provide criticism on learner performance, hints on the problem nature, or suggestions on how to proceed, (d) the option of having the expert module demonstrate optimal performance afterwards, and (e) the use of different ways of depicting the simulated system.

A selection of findings is summarized by placing them under the four themes we think to be characteristic of learning with computer simulations (see de Jong, this volume).

Education & Computing 6 (1991), 305-358
Elsevier

1. Introduction

The use of computer simulations in education and training can have substantial advantages over other approaches. For ethical, safety, or cost-benefit reasons an on-the-job training approach may be undesirable, and sometimes even impossible. In comparison to alternatives such as textbooks, lectures, and tutorial courseware, a simulation-based approach offers the opportunity to learn in a relatively realistic problem-solving context, to practise task performance without stress, to systematically explore both realistic and hypothetical situations, to change the time-scale of events, and to interact with simplified versions of the process or system being simulated (Hijne & van Berkum, 1990; de Jong, this volume; Alessi & Trollip, 1985; Merrill, 1987; Reigeluth & Schwartz, 1989; Schank & Farrell, 1988).

In view of these potential benefits, the results of computer-based training and education have been fairly disappointing (Goodyear et al., this volume). Learners are often unable to cope with the freedom offered by, and the complexity of, a simulation. As a result many of them resort to an unsystematic, unproductive mode of exploration.

There is evidence that simulation-based learning

can be improved if the learner is supported while working with the simulation (Goodyear et al., this volume). Part of this instructional support can be achieved through 'off-screen' written material (e.g. a workbook of exercises), and sometimes an individual coach is available to monitor and assist the learner at work. A large part of the necessary support functionality can, however, be accommodated within the computer program itself. The resulting system, what we call an Intelligent Simulation Learning Environment (ISLE), embeds a computer simulation within an instructional environment that aims to support the learning process in an adaptive way. The work reported in this paper has been carried out in the context of SIMULATE, a European Community project aimed at the development of an authoring workbench for Intelligent Simulation Learning Environments (Hijne & de Jong, 1989; de Jong, this volume).

Obviously, a learner can be supported in many different ways, some of which will be more appropriate than others. This paper reports on the results of an inventory we made to assess the various options for instructional support of simulation-based learning. The inventory has focused on various instructional measures to actively guide and direct the learner in exploring a simulation. The options for interface design, although equally relevant as means of support, are described elsewhere (de Hoog et al., this volume).

Although we cite evidence for the effectiveness of various instructional support options wherever available, we do not favour particular options over others (see van Berkum (1991) for a principled selection). The aim here has simply been to describe a wide range of available options along with their justification.

The appropriateness of various forms of instructional support strongly depends on characteristics of the simulation domain, on characteristics of the learner, and on the instructional goals one hopes to achieve. Whereas the first two issues have already received an in-depth discussion (van Joolingen & de Jong, this volume; Goodyear et al. this volume), the third has only been briefly touched upon (van Berkum et al., this volume). Before reporting on the instructional support survey, we will therefore first investigate the variety of instructional goals for simulation-based learning.

2. Learning goals for simulations.

Computer simulations are being used for a wide range of learning goals, such as radar system troubleshooting skills (Kurland, 1989), scientific inquiry skills (Shute & Glaser, 1990), or understanding of the cardiovascular system (Min, 1987). The design and use of an Intelligent Simulation Learning Environment (ISLE) obviously depends on the *learning result* being aimed at. This may seem a trivial statement, since different ISLE's will usually aim at knowledge of different domains. However, learning a particular qualitative physics model, for example, differs from learning how to carry out a correct aircraft landing procedure in the *type* of knowledge to be acquired, and not just the domain from which it is derived. The type of knowledge to be learned seems a very important factor in designing and using simulation-based learning environments.

Learning a qualitative physics model will probably be facilitated by an ISLE which includes instructional strategies and actions that induce 'deep' mental processing of the subject-matter at hand, learner modelling facilities which keep track of the learner's deep understanding, and an interface that represents the simulation model in a conceptually adequate (and not necessarily in a highly realistic) way.

Learning how to carry out a correct and swift landing procedure, on the other hand, will probably benefit more from an ISLE which includes instructional strategies and actions that induce increasingly automatic performance, learner modelling facilities which keep track of the learner's sequencing and timing of procedure steps, and an interface that has a high degree of physical realism.

The above examples suggest that the type of simulation learning goal to a large extent determines the type of ISLE to be used. In view of this, it is important to have a plausible *classification of learning goals for simulations*, one which captures relevant dimensions of the knowledge to be learned. This first part of the paper will work towards such a classification.

In order to narrow down our concept of a learning goal, we will first relate it to standard conceptions in the educational literature. Then, as there are

of instruction, such as 'get sufficient learner attention' or 'find misconception held by the learner'.

Furthermore, we are not particularly interested in *who* actually entertains certain learning goals. In educational situations one should usually distinguish between learning goals entertained by the instructional agent (such as a human tutor) and those entertained by the learner him- or herself. However, for present purposes it is not necessary to draw this distinction, since we are dealing with types of learning results that can in principle be achieved by educational simulations. Both agents in the educational situation may refer to their goals in terms of the proposed classification[1].

Finally, and at the risk of labouring the obvious, learning goals do not include such things as safety, inexpensiveness, or the possibility of reduced complexity. Such items express the belief that simulation can have specific advantages over other forms of instruction in achieving a particular learning goal. These items do not, however, address that particular learning goal itself.

2.2. *General taxonomies of learning goals.*

Instructional design theory has a long-standing tradition with respect to the classification of learning goals. Before getting into simulation-specific taxonomies, and in order to provide some relevant background, a few of these more general ones will be described. Two have been highly influential in the field of education (Bloom, 1956; Gagné, 1965), while a third and more recent one (Romiszowski, 1981; 1984) is a useful elaboration of its predecessors.

Bloom's learning goal tripartition (Bloom, 1956; Kratwohl, Bloom, & Masia, 1964) has recently been described as follows:

1. *Knowledge-related objectives*, i.e. "...related to the acquisition and application of knowledge and understanding." (Percival & Ellington, 1988, p. 55).
2. *Attitude-related objectives*, i.e. "...concerned with attitudes and feelings which are brought about as a result of some educational process." (pp. 55-56).
3. *Motor skills-related objectives*, i.e. dealing with "... the development of manipulative or physical skills." (p. 56).

Bloom himself referred to objectives in these classes as belonging to the *cognitive*, *affective*, or *psychomotor domain* respectively. His treatment of the cognitive domain (Bloom, 1956) subdivided cognitive learning goals as pertaining to either knowledge, comprehension, application, analysis, synthesis, or evaluation. Bloom's sequencing of these subdomains should be taken to reflect "... milestones on the way to perfect accomplishment, rather than watertight categories with specific and exclusive characteristics." (Percival & Ellington, 1988, p. 58). At the lower end, having achieved *knowledge* would simply allow a learner to for instance recognize an object or state a definition of it. At the 'higher' end of *evaluation*, learners would for instance be able to indicate logical fallacies in arguments, or compare different arguments.

A similar subdivision has been worked out for the affective domain (Kratwohl, Bloom, & Masia, 1964), but the psychomotor domain more or less escaped Bloom's attention (Romiszowski, 1984, p. 37). Although research attempts to validate both the cognitive and affective subdivisions have proved inconclusive, Bloom's global tripartition, as well as the formalization attempt by itself, have very much influenced later thinking about learning goals. With respect to simulation learning it would seem that all three types of goals can be of relevance.

A second influential classification of learning goals, or *learned capabilities* in his terms, has been proposed by Gagné and colleagues (e.g. 1965; 1985; Gagné & Glaser, 1987; Wager & Gagné, 1988). It is an attempt to provide a sufficiently detailed classification of learning goals, allowing for subsequent prescription of particular instructional strategies and actions (instructional *events*). Five types of learned capabilities are distinguished:

1. *Intellectual skill*, which "... enables the learner to do something that requires cognitive process-

[1]This is why we would also like to reserve the term *instructional goals* for goals entertained by the instructional agent, and *learner goals* for goals entertained by the learner.

ing. It is *procedural knowledge.*" (Wager & Gagné, 1988, p. 36).

2. *Cognitive strategy*, referring to "... skill by means of which learners exercise control over their own processes of learning and thinking." (p. 39-40).
3. *Verbal information*, "... sometimes called *declarative knowledge*, to imply that its acquisition makes it possible for the learner to *declare* it, or *state* it." (p. 40).
4. *Attitude*, involving an "... acquired internal (motivational) state that influences the learner's choice of personal action." (p. 41).
5. *Motor skill*, which involves refining (muscular) movement "... in timing and smoothness by the learning that occurs during practice." (p. 42).

Gagné's approach is closely related to information processing accounts of human learning. Particular instructional events should induce particular *internal learning processes*, which would in turn move the learner towards a particular learning goal. For our purposes, the major difference from Bloom's taxonomy involves an alternative partitioning of the cognitive domain into intellectual skill ('procedural knowledge'), cognitive strategy, and verbal information ('declarative knowledge'). If one momentarily adopts these distinctions, it would seem that simulation has more learning potential for intellectual skill and cognitive strategy than for verbal information. However, in a later section we will argue for an alternative and more explicit partitioning of the cognitive domain, more in line with current notions in cognitive psychology.

The final learning goal taxonomy to be discussed comes from Romiszowski (1981; 1984). Although up until now it has not been as influential as the previous two, it can be viewed as representative of the more recent approach to learning results (e.g. Williams, 1977; Merrill, 1983; 1988).

One of Romiszowski's objections to Bloom's tripartition is that it does not really map onto current ideas in industrial training contexts, particularly those involving the difference between knowledge and skill. In order to compensate for this mismatch, he attempts to draw the line more clearly:

"1. *Knowledge* refers to the information stored in the learner's mind.
2. *Skill* refers to actions (intellectual or physical) and to reactions (to ideas, things, or people) which a person performs in a competent way in order to achieve a goal." (Romiszowski, 1984, p. 41).

According to the author, *knowledge* can either be factual or conceptual, with factual knowledge being subdivided into facts and procedures, and conceptual knowledge into concepts and principles. This adds up to four types of knowledge, with quoted descriptions from Romiszowski (1984, p. 42):

1. *Facts*: "Knowing facts, objects, events or people (either directly as concrete experiences, or as verbal information)."
2. *Procedures*: "Knowing what to do in given situations (i.e. knowing the procedure)."
3. *Concepts*: "Specific concepts or groups of concepts (being able to define, etc.)."
4. *Principles*: "Rules or principles which link certain concepts or facts in specific ways (enabling one to explain or predict phenomena)."

Skills can be classified into four categories as well, which basically reflect Bloom's three domains plus one for human interaction:

1. *Cognitive skills.*
2. *Psychomotor skills.*
3. *Reactive skills*; reacting "... to things, situations or people in terms of values, emotions, feelings (self-control skills)." (p. 42).
4. *Interactive skills*; interacting "... with people in order to achieve some goal, such as communication, education, acceptance, persuasion, etc. (skills in controlling others)." (p. 42).

A major theme underlying Romiszowski's taxonomy is that different types of information (content) should not be confounded with different ways of using information (operations). This view implies that the concept of *skill* is not exclusively tied to knowledge of procedures. Since, as Romiszowski puts it, "procedures are in themselves information and the learning of the steps of a given procedure leads to the acquisition of knowledge." (p. 40), one

can have knowledge about procedures without being particularly 'skilled' in executing them.

Recent work on skill acquisition is clearly consistent with the above separation of skill from knowledge of procedures. Before getting into the relevant research, the next section examines a number of simulation-specific learning goal taxonomies.

2.3. *Taxonomies of learning goals for simulations.*

The increasing interest in using computer simulations for educational purposes has led to several recent attempts to classify the potential learning goals involved. In the current section, two of these efforts will be discussed. One taxonomy resulted from describing simulation types as part of a general overview of computer-assisted instruction (Alessi & Trollip, 1985), and a second one from applying a particular instructional design theory to computer simulation (Reigeluth & Schwartz, 1989).

In their overview of computer-based instruction, Alessi and Trollip (1985) devote a separate chapter to educational simulation. Although their classification of simulation types is definitely not the most consistent one around (van Joolingen & de Jong, this volume), it does to some extent reflect what the authors see as viable learning goals. We will therefore briefly review their classification with respect to the goals involved.

Physical simulation allows for the manipulation of physical objects displayed on the screen, "... giving the student the opportunity to use it or learn about it." (Alessi & Trollip, 1984, p. 162). The authors give the example of a flight simulator, by means of which the student can learn apparently superficial relationships between use of the controls, instrument readings, and the plane's passage through the air. Other examples involve the acquisition of 'deeper' model relationships (by simulating free fall experiments) and procedures for operating particular devices (such as a simulated Reverse-Polish-Notation calculator). The commonality in these examples seems to be the necessity of a directly manipulable physical object for learning, each time serving a different learning goal.

Procedural simulations are more directly linked to a particular goal, which involves learning "... about how the simulated machine works, not as an end in itself, but rather as a means for acquiring the skills and actions needed to operate it." (p. 165). Examples are landing a space shuttle, performing a titration, or diagnosing a patient. These can also be physical simulations; what makes them procedural, however, is the presence of a normative sequence of steps to be preformed.

Situational simulations, such as that of a teacher experiencing a first year of teaching practice, address Romiszowski's reactive and interactive domains. These simulations "... deal with the attitudes and behaviors of people in different situations, rather than with skilled performance." (p. 165).

Process simulations, finally, are distinguished from the above three by the fact that learner activity is restricted to initial condition setting, after which the independent development of a process can be observed. Examples, such as a simulation involving the balance between predator and prey populations, suggest that the learner is supposed to arrive at a 'deep' understanding of a complex and dynamic system.

To summarize, Alessi and Trollip explicitly mention the acquisition of system understanding (physical & process simulations), of action-sequences (procedural simulations), and of understanding and coping with situations involving other people (situational simulations). However, they are not particularly systematic in their approach to simulation learning goals. There is no explicit separation of knowledge and its skill-related aspects (Romiszowski), nor any reference to more domain-independent learning goals.

Recently, Reigeluth and Schwartz (1989) have tried to provide an instructional design theory for computer simulation, resulting in a somewhat more articulate classification of learning goals. The authors make the plausible assumption that, in view of its dynamic and interactive nature, simulation is "... an ideal medium for teaching student content that involves changes." (Reigeluth & Schwartz, 1989, p. 2). Following this assumption, and adhering to the often used fact-concept-principle-procedure classification (Merrill, 1983; Romiszowski, 1981), they restrict their attention to the use of simulations for learning about principles and pro-

cedures.

The authors differentiate three types of learning goals, each one defining a particular simulation type:

1. A *procedural simulation* is used to "... teach the learner to perform a sequence of steps and/or decisions." (p. 2), e.g. procedures for flying an airplane or writing a good paragraph.
2. A *process simulation* "... teaches naturally occurring phenomena composed of a specific sequence of events." (p. 2), such as the process of photosynthesis.
3. A *causal simulation* "... teaches the cause-effect relationship between two or more changes." (p. 2), e.g. as involved in the law of supply and demand.

In line with current ideas on the separation of knowledge and its use, Reigeluth and Schwartz also discriminate between an acquisition and an application phase in the learning process. In the *acquisition phase* the learner must acquire an understanding of change relationships (process/causal principles), or knowledge of what steps to follow when and how (procedures). In the *application phase*, further learning involves the generalization of acquired principles or procedures, the successful utilization of (causal) principles in for instance prediction, and the development of automaticity in carrying out procedures.

Incidentally, the authors state that the latter phase need not necessarily involve such automatization. This means that the learning goal for procedure execution may be reached after having acquired knowledge about the procedure, and one need not aim at skill. The issue was already raised in the context of Romiszowski's taxonomy: knowledge of procedures is not exclusively tied to skill, but may be a proper learning goal by itself.

According to Reigeluth and Schwartz, process simulations deal with naturally occurring events, while causal simulations address the relation between two or more changes. However, this difference does not seem to be a basic one, since a process simulation also addresses the relation between two or more changes: a time-dependent system on the one hand, and passing time on the other. It remains to be seen whether, as the authors propose, both types of simulation really require different instructional environments.

Although the authors acknowledge the existence of considerable overlap with the Alessi & Trollip taxonomy, the present classification of learning goals is certainly more articulate than the previous one. It indicates that certain kinds of knowledge can be involved, and also refers to its skill-related use as a separate issue. Again, however, the possibility of relatively domain-independent learning goals is not considered. Moreover, the authors do not mention the use of simulation for affect-related objectives.

In our view, the restriction to principles and procedures as two major kinds of knowledge is an unnecessarily strong one. Simulation can also be used to learn about concepts (particularly when they involve change, e.g. 'acceleration' or 'decay') and even facts (e.g. the function of specific system components). As will be seen below, we therefore propose to distinguish between *operational knowledge* (procedures) on the one hand, and *conceptual knowledge* (principles, concepts, facts) on the other.

2.4. *Dimensions of a simulation learning goal.*

In order to justify our proposal for yet another classification of simulation learning goals, we will have to show why previous taxonomies are not adequate for the design and use of intelligent simulation learning environments (ISLE's). The major drawback of existing taxonomies appears to involve the issue of knowledge and skills, and its relation to the procedural-declarative distinction. Although it has already received some attention in previous sections, the issue will now be elaborated further, leading to a proposal similar (but not equivalent) to Romiszowski's.

2.4.1. Knowledge, skill, and the procedural-declarative distinction.

Looking back at the general and simulation-specific learning goal taxonomies discussed, one finds two different views on the relation between skill and the type of knowledge it involves. Sometimes, skill is more or less equated with knowledge of procedures

(particularly by Gagné, but also by Alessi and Trollip). This is usually expressed in the phrase 'procedural skill', opposing it to knowledge of principles, concepts, and facts (which is then usually called 'declarative knowledge'). On the other hand, some authors (particularly Romiszowski, but also Reigeluth and Schwartz) have tried to dissociate knowledge and its skill-related aspects from the issue of whether that knowledge concerns principles, concepts and facts on the one hand, or procedures on the other.

Below, we will argue that it is necessary to separate a dimension of *how knowledge is represented* (declarative vs. procedural or 'compiled') from *what that knowledge is about* (e.g. procedures or principles). After briefly reviewing the original meaning of the procedural-declarative distinction we will discuss recent cognitive science work on skill acquisition, and derive the implications for a simulation learning goal taxonomy.

The basic issue involved in the procedural-declarative distinction has been nicely described by Rumelhart and Norman (1981, p. 336-337):

> ...it is impossible to evaluate a representational system apart from the process that operates on it. Consequently, in modeling any cognitive process, there is always the problem of deciding how to partition that part of the knowledge system that is "process" from that part that is "data." Depending on the relative amounts of the system allocated to "process" and to "data," we have what Winograd (1975) has called "procedural" or "declarative" representational systems. Some authors have emphasized the "data," trying to have as few special-purpose procedures as possible; such as system is called *declarative*. Others have emphasized the processes involved and have largely embedded the knowledge of the system within these processes. These systems are generally called *procedural*.

Rumelhart and Norman continue to summarize the differences between these two types of representation: whereas a declarative representation would allow for more *flexible system behaviour*, for *easier acquisition* of new knowledge, and for *easier access* to knowledge already stored, a procedural representation would sacrifice these three benefits in order to support more *efficient performance*.

The above characterization of differences between procedural and declarative representation is stated in very general 'representational systems' vocabulary. However, there is good evidence that something of the kind is also involved in *human* cognitive and noncognitive performance. That is, it appears that knowledge encoded in the human system is sometimes relatively easy to acquire, and once acquired very accessible to introspection and flexible in its use (declarative-like). At other times, knowledge is only learned with much practice, and once acquired it is much less accessible to introspection and flexible in use. All this, however, is compensated for by an increase in performance automaticity and efficiency (procedural-like)[2]. Much of the evidence involved for both representation types comes from studies of novice-expert differences (see Anderson, 1985, pp. 232-260 for an overview), and of one of the processes that turn the novice into an expert: skill acquisition.

Anderson's influential theory of skill acquisition (e.g. 1983; 1985; 1987) suggests that practice in performance on some task leads to a gradual shift in the knowledge representation used in that performance. In line with earlier theories (particularly Fitts, 1964; Fitts & Posner, 1967), Anderson distinguishes three stages in the acquisition of a particular skill:

1. a *declarative stage* in which "... skill-independent productions use declarative representations relevant to the skill to guide behavior." (Anderson, 1983, p. 216),
2. a *knowledge compilation stage* by which "... the system goes from this interpretive application of declarative knowledge to procedures (productions) that apply the knowledge directly." (pp. 231-232)[3], and
3. a *tuning stage*, in which the 'compiled' knowledge is refined and consolidated through further practice.

---

[2]It is *not* our intention to say anything definite on the precise nature of human knowledge representation, such as "procedural knowledge definitely involves productions". What we *do* want to convey is that, no matter the underlying causes, there may be differences in the ease with which particular knowledge can be acquired and used (expressed in terms of flexibility, learnability, accessibility, and efficiency).

[3]Here, the use of concepts like 'interpretive processing' and 'compilation' is very similar to the original computer science use (e.g. when talking about Lisp).

Skill acquisition is by no means exclusively a perceptual-motor phenomenon. Both Anderson (e.g. 1985) and others (e.g. Newell & Rosenbloom, 1981) provide many examples where the skills acquired are purely cognitive, such as generating geometry proofs or developing computer programs.

More importantly, the difference between novices and experts seems to involve more than just a different representation of knowledge of *procedures*. As Anderson (1985) discusses in an overview chapter on the topic, the development of expertise in a particular domain does not only rest on the development of more efficient and appropriate problem-solving procedures, but to a large extent also depends on the perception and representation of larger, and more meaningful chunks of problem features. The importance of this latter development has received its classic demonstration in the domain of chess (e.g. de Groot, 1965). Chess masters mainly differ from novices in their 'direct perception' of complex, meaningful chess patterns, and much less in their basic problem-solving procedures. In Anderson's words, "... masters effectively "see" possibilities for moves; they do not have to think them out." (1985, p. 245).

The chess example shows that skill and skill acquisition are not necessarily tied to the development of expertise in carrying out procedures (*operational knowledge*), but also seem to involve qualitative changes in the perception and representation of more *conceptual knowledge*, such as the relation between different pieces in a chess pattern. This suggests that the phenomenon of knowledge compilation is not limited to operational knowledge, but that it also extends to more conceptual knowledge (particularly principles, and concepts that involve change).

Thus, while recent work in instructional design theory (e.g. Romiszowski) suggests a dissociation between skill and the kind of knowledge it involves, cognitive science research now underpins such propositions. Skill acquisition appears to involve a *qualitative change* in the representation of knowledge used in performing that skill, with an increasing reliance on compiled rather than declarative representations[4]. As novice-expert studies have shown, these representational changes can pertain to *both* operational and conceptual knowledge[5].

Note, however, that the above interpretation differs from Romiszowski's in a subtle way. While the latter conceives of skills as efficient means for *applying* declaratively represented knowledge to a task, we see them as *incorporating* this same knowledge in a compiled format. For this reason, the distinction we propose is not between knowledge and skill, but between declarative and compiled representations of knowledge. In our view, skill involves, among other things[6], an increasing reliance on compiled knowledge, which has its own advantages and drawbacks. The intended separation between the *kind of knowledge* (conceptual or operational) and its *representation* (declarative or compiled) is illustrated in Figure 1.

Some choices implicit in this scheme need to be made explicit. First of all, the affect-oriented domain (Bloom, Gagné, Romiszowski) has been left out. Simulations may be very useful in inducing a different attitude, for example, when simulating the effect of energy consumption behaviour on the planet's resources. The attitude involved may also be one toward one's own (actual or expected) performance, for instance when using simulation to increase confidence. We do acknowledge the poten-

---

[4]Some interesting work by Broadbent and colleagues (e.g. Broadbent & Aston, 1978; Hayes & Broadbent, 1988) suggests that the acquisition of compiled knowledge need not always be mediated by a declarative stage. Learning to control a complex dynamic system will often take place in an environment with very much information as such (variables and relations), but with insufficient information as to what is truly relevant. In such task environments, learners frequently seem to engage in a mode of 'implicit' or 'unselective' learning, which, according to Hayes and Broadbent (1988), directly leads to a more compiled representation of knowledge.

[5]To avoid confusion, we will refrain from using the familiar but ambiguous term 'procedural knowledge' in our taxonomy. As argued, this term hides the meaning of *operational knowledge* (procedures) and *compiled knowledge*, and also erroneously suggests that the two always go together.

[6]Payne (1988) has likewise argued that the traditional, ACT*-like view of skill *as method* is a too restrictive one, since it denies the important role played by conceptual knowledge in the development of expertise. According to the author "...there are other skills besides purely procedural ones...", "...even skills with a large procedural component rely on a great deal of conceptual knowledge..." (Payne, 1988, p. 76).

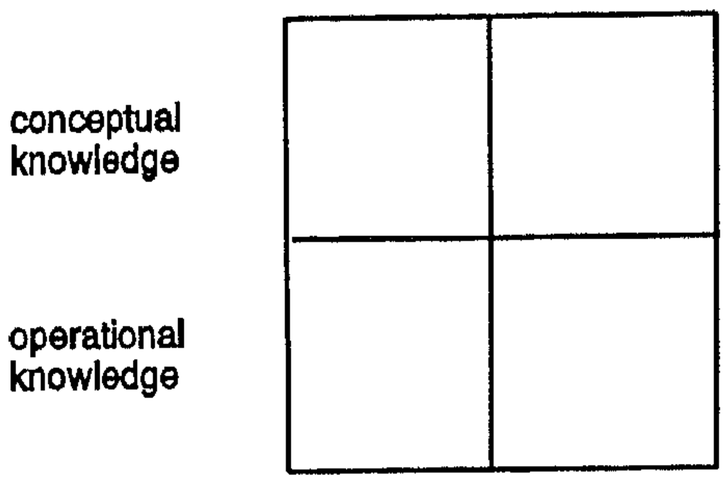

Figure 1 *Classification of simulation learning goals with respect to the kind of knowledge (conceptual -- operational) and its representation (declarative -- compiled).*

tial of simulations in the affective domain, but we have nevertheless chosen to focus on 'colder' cognition for now.

Secondly, although we agree with Reigeluth and Schwartz (1989) that simulation-based learning will be particularly useful for subject matter that involves changes, we do not wish to exclude its potential for learning about concepts and facts as well. We have therefore merged the latter two types of knowledge with principles into a single category (conceptual knowledge). The fact that we have done so also reveals our assumption that, in terms of design implications for instructional simulation, the important distinction lies between principles, concepts and facts on the one hand and procedures on the other.

Thirdly, although the psychomotor domain (Bloom, Gagné) may appear to have been ignored, we take operational knowledge to pertain both to internal (cognitive) and *external* (perceptual-motor) behaviour. High fidelity of the simulation display and of input devices (e.g. a realistic control yoke for flight simulation) can in principle support the acquisition of properly sequenced and smoothly executed perceptual-motor behaviour.

The expressive power of the scheme in Figure 1 has been hinted at a few times already. In the limited conception which equates declarative knowledge with knowledge of principles, concepts and facts, and compiled knowledge with knowledge of procedures, simulation learning goals would only fall in either the top-left or the bottom-right cell. In the current approach, however, we have two new cells which may well involve simulation learning goals interesting in their own right.

The bottom-left cell points to what can be referred to as *declarative operational knowledge*. In the area where simulation may be of use, this could be a desirable learning goal in, for instance, situations where systematic troubleshooting must be learned, but where the knowledge of troubleshooting procedures must remain explicit and accessible. Although this is a perfectly sound learning goal, we acknowledge that it may not really capitalize much on the specific advantages of simulation over other forms of instruction.

On the other hand, the top-right cell seems to point to a much more interesting use of simulations. This cell designates what one may call *compiled conceptual knowledge*; perhaps this is the type of knowledge involved in a phrase like 'getting a feel for the underlying model' (without being able to articulate its internal workings). In any case, it seems related to the notion of 'implicit' or 'unselective' acquisition of relations, which, according to Hayes and Broadbent (1988), often occurs when learning to control a complex dynamic system. It also seems related to the previously discussed 'immediate perception' of domain-experts. If one considers that such experts have spent time on thousands of problems in their field before acquiring some immediate perception of relevant cues and meaningful patterns, simulation learning may have much to offer here[7].

2.4.2. Domain-independence.

Knowledge acquired in a particular subject-matter domain may be of use in other domains because the structure of these domains resembles the structure of the original one. As an example in the context of simulation, consider the learning of system troubleshooting heuristics. Such heuristics may initially be specific to the system at hand (e.g. an amplifier of type X), but some of them may *generalize across similar domains* (e.g. all electronic systems). In its extreme form, generalization across domains may result in highly generic problem-

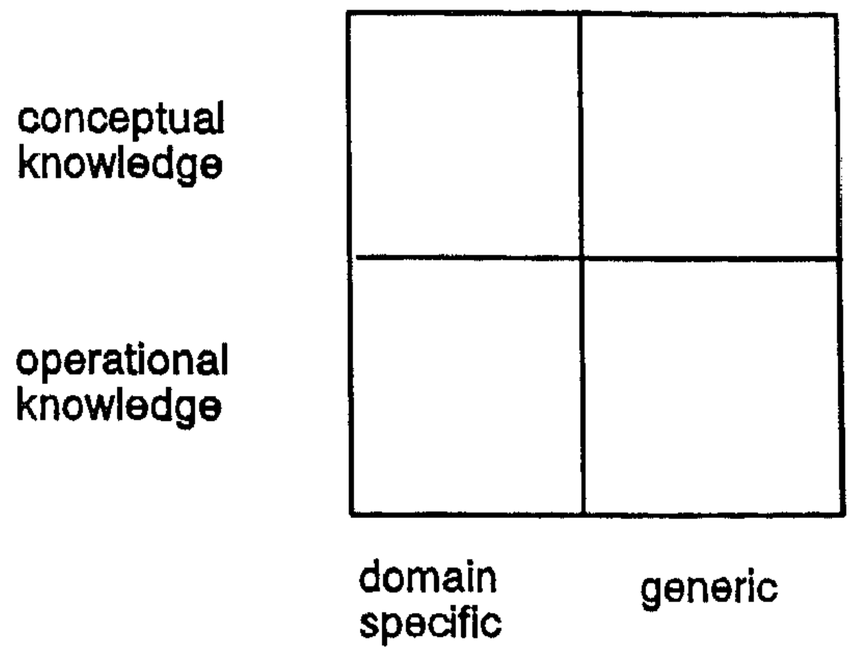

Figure 2 *Classification of simulation learning goals with respect to the kind of knowledge (conceptual -- operational) and its scope (domain-specific -- generic).*

solving heuristics such as means-end analysis or breadth-first search. This kind of example deals with operational knowledge, and is very much related to Gagné's cognitive strategy objective (see also Gagné & Glaser, 1987, pp. 67-68).

Another example, addressing conceptual rather than operational knowledge, involves the generalization of particular domain-related principles into knowledge about the general structure of a dynamic system, or about the exponential shape of many laws of nature. Of course, a simulation system may facilitate generalization across domains by means of multiple simulations pertaining to different domains, and by emphasizing the generalities involved.

In order to accommodate the above differences, we distinguish the *scope of knowledge* (domain-specific vs. generic) from the *kind of knowledge*. This is illustrated in Figure 2 (left). Of course, this scope dimension is a continuous one. Moving along this continuum reflects increasing or decreasing degrees of generalizability, or *transfer* of knowledge.

2.5. *Learning goals for simulations - towards a new taxonomy.*

2.5.1. The learning goal cube.

This paper has reviewed several general and simulation-specific learning goal taxonomies, and has argued for a more articulate description of the relevant dimensions involved. To summarize briefly, we have argued for three orthogonal dimensions on which a particular learning goal can be classified:

1. The kind of knowledge to be learned may be conceptual or operational. *Conceptual knowledge* is knowledge of principles, concepts and facts related to the (class of) system(s) being simulated. *Operational knowledge* is knowledge about sequences of cognitive and/or noncognitive operations (procedures) that can be applied to the (class of) simulated system(s).
2. Knowledge can be encoded in a declarative or a compiled representational format. *Declarative knowledge* is represented in a format that is relatively easy to acquire, that makes the knowledge relatively easy to report upon, that makes the knowledge of potential use in an unlimited number of problem contexts, and that requires interpretation in order to use it in a task. *Compiled knowledge*, on the other hand, is represented in a format that is only obtained after using the knowledge in a problem-solving context, that makes the knowledge hard to report upon, that restricts its potential use to a limited number of contexts, and that

[7]It seems plausible that this type of learning goal will benefit from an ISLE which presents *many relevant and irrelevant variables as they occur in the natural environment* to the learner. This runs against the idea that simulation is often useful because natural complexity can be reduced. The issue may be illustrated with two examples from the medical domain. If one aims for 'deep' understanding of the cardiovascular system (declarative knowledge of cardiovascular principles), it seems a good idea to at least start out with a reduced simulation environment, only dealing with a few relevant cardiovascular variables (e.g. CARDIO (Min, 1987)). However, if one aims at the ability to 'directly perceive' complex patterns of symptoms, which allows for a quick zooming in on the proper medical diagnosis (compiled knowledge of diagnostic principles), the learner may benefit most from a very complex simulation involving many variables, only some of which are relevant to any particular syndrome (e.g. HUMAN (Coleman & Randall, 1986)). These considerations in fact illustrate the relevance of a learning goal typology for the design of a simulation-based learning environment.

can be used in a more automatic, effortless way.

3. The target knowledge may finally vary in scope. *Domain-specific knowledge* is specific to the simulation domain at hand. *Generic knowledge* is not specific to the simulation domain at hand, but extends to other domains as well.

Although both the second and third dimension are continuous in the sense that some particular knowledge can be *relatively* compiled or generic, we can depict the space of learning goals for simulation in a 3-dimensional matrix (Figure 3).

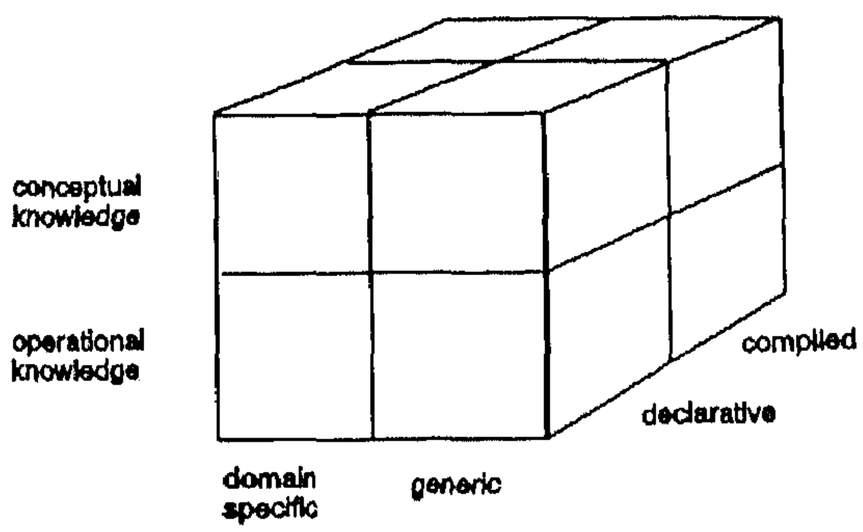

Figure 3 *Classification of simulation learning goals with respect to the kind of knowledge, its scope, and its representation.*

A few examples may illustrate the use of this matrix, or 'goal cube'. As a first one, imagine a course in general systems theory, in which the learners interact with three simulations from different domains (say, electronics, economics, and biology), in order to acquire an explicit understanding of a negative feedback relation between dynamic system variables. This learning goal involves knowledge of a relation (conceptual knowledge), which is not specific to the three domains employed (relatively generic), and which should be easily accessible for verbal reporting and use in different contexts (declarative representation).

As a second example, take a simulation training course in which nuclear power station operators should learn how to react to cooling system failure in a quick and automatic way, and without making any mistakes due to stress, panic, etc. Such a learning goal involves knowledge of a normative sequence of actions (operational knowledge), which is specific to the system being simulated (domain-specific), and which supports fast, robust, and automatic performance at system failure time (compiled representation).

A final example may be the use of a highly complex simulation of the human physiological system in order to train students in medical diagnosis, and particularly in the 'immediate' recognition of complex symptom patterns among many potential disease cues. Here, the learning goal involves knowledge of common relations between symptoms (conceptual knowledge), which is only valid for the human system (domain-specific), and which supports relatively automatic pattern matching performance (compiled representation).

While the relevance of each dimension has been argued for, and examples have been given throughout the paper, it is not our intention to come up with typical examples for each of the eight cells making up the matrix. Not all cells may be of equal significance to educational simulation. However, some cells may point to less obvious but nonetheless interesting goals for simulation (e.g. compiled representation of domain-specific conceptual knowledge). Of course, a simulation need not be restricted to a single cell (or, in dimensional terms, to a single point in the goal space). It is in fact very likely that any educational simulation will aim at multiple types of learning goals, although some of those may be regarded as interesting side-effects, or may not be explicitly acknowledged at all.

Applying the above taxonomy to 'real life' curricular situations will not always be very straightforward. Any particular cluster of performance aimed at in simulation learning may rely on both conceptual and operational knowledge, part of which may be domain-specific, and part of which may be relatively generic. Likewise, a performance cluster may rely on knowledge which is partly represented in declarative format, and partly stored in a more compiled way. The fact that it may be difficult to disentangle all this in 'real life' is probably more an indication of life's complexity than of taxonomical validity.

2.5.2. Learning about knowledge acquisition.

The learning goals covered by the cube all involve knowledge about particular simulation domains (or classes of simulation domains). However, we have ignored a very plausible class of simulation learning goals, which address the process of *knowledge acquisition* itself. Consider, for instance, using a simulation to teach people how to efficiently obtain and validate knowledge about a system through experimentation[8]. To some extent, this will involve relatively generic knowledge of systems structure and function (conceptual knowledge), and of how to deal with systems (operational knowledge). Clearly, however, learning about knowledge acquisition is a kind of second-order learning: it involves knowledge about *one's own cognitive processes and products*.

There is evidence that unsuccessful learners differ to a large extent from successful ones in the amount and quality of such second-order knowledge (e.g. Gagné & Glaser, 1987; Glaser & Bassok, 1989). In this context, the term *metacognition* is often used to indicate knowledge about one's knowledge (e.g. whether a belief is warranted by evidence, or what important knowledge gaps need to be remediated), whereas the terms *executive control* and *self-regulation* usually refer to knowledge of related procedures (e.g. for the allocation of one's cognitive resources and time, the verification of conclusions drawn, or of one's learning results). All this seems particularly vital to successful exploration-based learning, and an ISLE could be very useful in fostering the development of one's knowledge acquisition capabilities.

However, it is not entirely obvious how to extend the above, domain-oriented taxonomy in order to accommodate for such metacognitive learning goals. One could imagine a second, *similar* cube for their classification (which would in fact be equivalent to the addition of a 4th dimension). The argument would be that learning about knowledge acquisition processes (a) can involve both conceptual knowledge (e.g. the relation between theory, hypothesis, and prediction) and operational knowledge (e.g. how to test an hypothesis), (b) can be represented in a relatively declarative or compiled way, and (c) can be more or less specific to the simulation domain at hand. While these speculative statements are by no means implausible ones, they should be subjected to further study. For the time being, we simply place this type of metacognitive learning goal (that is, those related to exploration-based knowledge acquisition) as a homogeneous class next to the eight learning goal types that make up the cube.

3. Instructional environments for simulations: introduction

To a large extent, the concept of an Intelligent Simulation Learning Environment (ISLE) rests on the assumption that disadvantages of *completely free* exploratory learning can be diminished or eliminated by imposing *instructional constraint* on the learning situation (see Section 1). Obviously, there is a tension here, since too much constraint can turn an exploratory simulation into a plain tutorial or drill-and-practice experience. One of the major challenges is to strike the right balance between exploratory freedom and instructional constraint, trying to get rid of some of the weaknesses without losing the strengths of simulation.

In the remainder of this paper we will describe the kinds of instructional constraint one could impose on simulations. This means that an inventory is made of all kinds of instructional 'approaches, strategies, and actions' that may be applicable to simulations or, more general, to exploratory learning environments. In order to deal with the enormous amount of possibly relevant literature in 'constrained exploratory' fashion, several heuristic questions can be asked in advance:

1. *What instructional features of potential relevance to exploratory learning can be derived from cognitive theory?* Current knowledge on how people learn, or how they solve problems in exploratory situations, can suggest potentially relevant instructional features.

[8] In fact, such a system has already been developed. In a later section we will describe Smithtown, a simulation-based learning environment for the acquisition of "scientific inquiry skills" (Shute & Glaser, 1990).

2. *What instructional features of potential relevance to exploratory learning are suggested by Instructional Design theories?* Instructional features can to some extent be derived from existing ID theories, and sometimes this derivation has already been done explicitly (e.g. for simulation learning in Reigeluth & Schwartz, 1989).

3. *What instructional features have been used within existing exploratory learning systems, and to what effect?* This question addresses the literature on exploration-based intelligent tutoring systems (ITS)[9]. 'Existing' should therefore be interpreted liberally, that is, also referring to systems still in the laboratory. If one would only take it to refer to the education and training field, there wouldn't be much ITS literature left.

It is impossible to provide an exhaustive overview of relevant work in the above areas, and our approach will necessarily be one of illustration. We will try to describe some of the more representative efforts in a particular area, but the selection of material will to some extent be arbitrary.

We will discuss the instructional support inventory in two different ways. Section 4 describes the material we found in relation to each of the above three questions (cognitive theory, Instructional Design theory, and exploration-based ITS's). Section 5 reorganizes this material around four central themes of simulation-based instruction (de Jong, this volume), and adds some new material that did not really fit in with any of the three questions posed above.

Before getting into the inventory material, our use of instructional concepts must be made explicit. First of all, we follow the distinction often made in literature between two kinds of *instructional measures*. The first kind, *instructional actions*, refers to measures defined at the actual interaction level. They include amongst (many) others: posing questions (of different types), providing examples, alternatives, explanation, etcetera.

Experienced teachers, however, do not solely act and react to the situation on a moment-to-moment basis. Rather, they coordinate several actions into higher-level *instructional strategies* in order to achieve a particular desired state of affairs (e.g. Leinhardt & Greeno, 1986). For instance, in order to facilitate acquisition, the teaching of new knowledge often involves referring to related knowledge already acquired before actually presenting the target information. Strategies like these are plans for achieving a particular aim through the coordination of instructional actions. These strategies have the function of *control mechanisms*: they control what is being presented and when.

In general, skillful instruction seems to involve some kind of strategic planning. At the highest level, such planning will be guided by general *instructional approaches*, which are often phrased in terms of desired learning type(s). So, for example, learning by observation is a general instructional approach, coaching is such an approach (the student is not guided through the domain in a structured way, but is monitored and corrected by a coach) and learning by instruction is such an approach (the student is guided through the domain in a structured way). Michalski (1987) lists some of these general approaches in the context of machine learning. Unfortunately, the distinction between an approach and a high-level strategy is not as clear-cut as one would like. In addition, *instructional principles*, such as 'minimize the learner's working memory load', capture common rules of thumb which can be implemented in particular features of the environment. Such principles often 'belong' to one or more instructional approaches, but can also be proposed in isolation.

To summarize, we will use the concept of instructional measures to refer to either instructional strategies or actions, the concept of instructional principles to indicate particular rules of thumb, and the concept of instructional approach to indicate the overall policy involved. Since the instructional environment may have features which are 'just there', and less related to active behaviour of the instructional agent, we will use the umbrella term *instructional features* to capture both these more passive features of the environment as well as the

[9]Next to ITS, the use of simulation within traditional CAI would also be of interest here. In view of the lack of scientific literature on simulation-based CAI, however, we will concentrate on ITS.

active measures taken by the instructional agent.

Literature shows a wide variety of terms related to instructional measures, such as instructional tactics (e.g. Ohlsson, 1986), events (e.g. Gagné, 1985), treatments (e.g. Gropper, 1983), rules (e.g. Landa, 1983), or methods and techniques (e.g. Percival & Ellington, 1988). This wouldn't be so bad if one could consistently map one terminology onto another, but this is usually not the case. What one author refers to as a tactic may be a strategy, action, or even a principle to someone else. We will therefore as much as possible refrain from using such related terms.

4. Instructional environments for simulations: A survey

4.1. *Cognitive theory.*

Instructional guidelines as used in CAL-systems or ITS's tend to have a rather eclectic character (examples of this can be found in Towne et al., 1988 and Reigeluth & Schwartz, 1989). The predominant basis for these guidelines, however, can be found in *cognitive theory*.

The rise of cognitivism has led to an enormous increase in the number of studies related to human cognition. Characteristic for this paradigm is:

a. Attention to non-observable processes and knowledge states;
b. A detailed study of expertise as compared to novice behaviour;
c. Revival of research methods such as thinking aloud, and
d. The study of complex, real life tasks additional to highly structured artificial tasks in a lab situation.

One of the main interesting results of cognitive research is the recognition of the existence of different *types of knowledge*. The main distinction is between conceptual knowledge and operational knowledge, a distinction that was already discussed in Section 2.4.1. More specific refinements are also made. In conceptual knowledge, *situational knowledge* (de Jong & Ferguson-Hessler, 1986) or *conditional knowledge* (Alexander & Judy, 1988) are sometimes distinguished. "Conditional knowledge entails the understanding of when and where to access certain facts or employ particular processes" (Alexander & Judy, 1988, p. 376). Within operational knowledge a subdivision is made for *strategic knowledge* (...the strategic processes that are responsible for the organization, coherence, and general execution the performance...., Gott (1989, p. 98); see also de Jong and Ferguson-Hessler (1986)). The classifications and refinements from the literature quite often contradict each other. A general conclusion, however, is that expert behaviour mostly consists of a delicate composition of all kinds of knowledge, and for some tasks some types of knowledge are more distinctive than others, thus calling upon different instructional strategies.

Despite the fact that most cognitive research is not directly aimed at studying the effects of instructional measures, the results in this field bear a direct relevance for instruction: cognitivism stressed the awareness for evolving knowledge of the learner (including prior knowledge) from a novice to an expert state and the importance of the learner's activities in this study process.

Glaser and Bassok (1989) presenting an overview of the relation between cognitive theory and instruction, indicate three areas in which the main accomplishments of cognitive theory have serious implications for instruction. These are, using the words of Glaser and Bassok:

a. knowledge organization and structure,
b. functional, proceduralized knowledge and skill, and
c. self-regulatory skills and performance control strategies.

These three fields have a direct relation to the three main types of instructional goals we have distinguished for learning with computer simulations (see Section 2). We will relate the topic of knowledge organization to the instructional goal of conceptual knowledge, for functional knowledge we will look at operational knowledge as a learning goal, and for self-regulatory skills at the knowledge acquisition goals. We will therefore follow Glaser and Bassok (1989) and relate their results to our

topic: learning with computer simulations. In this we will frequently touch upon research performed in the ITS field, because this is the area where cognitive theory has found its major 'test bed'.

4.1.1. Conceptual knowledge

One of the aims of learning with computer simulations can be knowledge of the *model* that underlies the simulation, or *conceptual knowledge* as we have called it (see Section 2 or van Joolingen & de Jong, this volume). In the context of simulations these models can be very complex (see Hartog, 1989).

Cognitive theory takes as one of its central aspects the internal model the learner creates of the information s/he is dealing with. This internal (mental) representation is a melding of the information in the 'outside world' and the prior knowledge in the long term memory of the learner (Gentner & Stevens, 1983). A summary of differences between experts and novices in knowledge and knowledge organization shows the following:

a. The most general finding is that experts organize their knowledge differently from novices. Experts tend to construct *chunks* of knowledge, that eventually become *schemata*. A schema is an organized body of knowledge, that is directly applicable to a certain situation. A schema contains *slots* for specific information. An expert has *default values* for these slots and in this respect the schemata help him/her to make sense of his/her environment and to react quickly and accurately to situations if necessary. Some studies, however, find that novices who are performing well also benefit from a schema based knowledge organization.

   The knowledge organization of novices is characterized by relations that are based at *superficial* characteristics of the elements of knowledge.
b. Novices tend to have inconsistencies in their knowledge without being aware of it and the knowledge of novices is characterized by the presence of bugs or misconceptions. Studies reporting misconceptions in a specific domain are numerous; studies that try to give a general description of misconceptions are sparse.

From these two findings we can conclude that learning is seen as a process of mental model transformation, in which the models get better organized, more coherent, more specific and contain fewer contradictions and misconceptions.

Some instructional principles can be used to guide this process[10]:

a. *Sequence and choice of models and problems*
   Offering models in a simulation in a specific order, such that relatively 'easy' models are offered first and the 'target' models last, is an instructional measure that explicitly takes into account the evolving nature of mental models. Such 'progressive' offering of models can be done in two different ways:
   - By providing a transition from *qualitative* to *quantitative* models or
   - By increasing the *number of elements and relations between elements* in a model.

   An excellent (and much cited) example of offering a sequence of models can be found in the work of White and Frederiksen (1987a,b; 1989; 1990) on QUEST. QUEST is an ITS based on a simulation of electrical circuits that helps students to learn troubleshooting in electrical systems. In QUEST learners are offered electrical circuits to explore. As learning progresses the models develop in the above-mentioned ways; White and Frederiksen say that the *order* and the *degree of elaboration* of a model increases.

   A third way in which models may differ in QUEST is in the *perspective* on a presented model. The models in QUEST have three perspectives:

   - *Functional perspective.* The model and its subsystems are described according to their purpose.
   - *Behaviourial perspective.* This is a view on the system at the level of circuit components.
   - *Physical perspective.* This refers to the

[10]Most of the principles listed here are taken from Glaser and Bassok who describe the work of White and Frederiksen (1987) on QUEST.

behaviour of the circuit at the level of physics.

While learning with QUEST learners are confronted with models that change from a qualitative to a quantitative nature, that are more elaborated and that go from a functional to a physical perspective. In this respect the instructional sequence follows the (assumed) transition from a novice knowledge state to an expert one.

b. *Providing explanations*
Providing students with explanations on the behaviour of models may help to cure inconsistent knowledge in learners. In QUEST the student can ask for *causal explanations* of circuit behaviour.

c. *Minimization of error*
Instead of remediating incomplete knowledge and misconceptions in learners, one can try to prevent these misconceptions from occurring. This is the approach taken in QUEST. The progressively more complex models in QUEST are offered to the learners in such a way that it is assumed that learners can master the model offered without making errors.

An important related principle is the principle of 'mastery learning', which means that models are always introduced that are close to the learner's mental model, which means that instruction uses an idea about the prior knowledge of the student (see also Hartog, 1989).

4.1.2. Operational knowledge

The second instructional goal that we distinguished was associated with the learning of procedures or skills, the input/output process related with the underlying model. The predominant theory in this respect is the ACT* theory by Anderson (Anderson, 1983). In his ACT* theory Anderson argues that skills develop from declarative knowledge with a general interpretation mechanism to specific procedures (functional knowledge as Glaser and Bassok call it). Productions take the form of condition-action pairs, that in the learning process (through a mechanism called compilation), become more domain specific and efficient. Although we adopt an approach slightly different from the one by Anderson and regard declarative (conceptual) knowledge and procedural (operational) knowledge as existing side by side (see Ferguson-Hessler & de Jong, 1990), the ideas about skill development and the combination of subskills are very useful.

The instructional principles from Anderson's ACT* theory are most clearly embodied in Anderson's LISP tutor (Anderson & Reiser, 1985). Glaser and Bassok (1989, see also Wenger, 1987) summarize these instructional principles as follows:

a. *Learning takes place in a problem solving context*
This approach is consistent with the general idea that we presented for learning with computer simulations. For Anderson problem solving is the only context in which declarative knowledge can be transformed into procedural knowledge (or operational knowledge as we call it).

b. *Model tracing*
In the philosophy of the ACT* theory an ITS contains an explicit model of the target behaviour (compiled procedural (operational) knowledge) in the form of production rules. An agent (student or system) will solve a problem by applying a number of production rules subsequently, thus travelling through the problem space. Next to the 'ideal knowledge' the LISP tutor also contains a large number of buggy rules. These buggy rules help in classifying the (incorrect) problem solving behaviour of learners. In a learning session, the LISP tutor matches the learner's behaviour to the ideal behaviour and uses this to provide feedback.

c. *Immediate feedback*
One of the principles used in ACT* based tutors is the principle of immediate error correction. When a learner takes a wrong path, s/he is immediately interrupted and given a hint about the direction in which to proceed. The idea is that immediate feedback provides the opportunity to point directly at the cause for taking a wrong path in the solution. It also avoids lengthy explanations that might be necessary if

the learner should be allowed to proceed down a wrong path.

d. *Minimization of working memory load*
The last principle mentioned by Glaser and Bassok (1989) in relation to Anderson's work is the principle of minimizing the working memory load. While learning, high demands are placed on the working memory, which implies that all tasks that are not instrumental to learning and occupy space in working memory are to be supported. In this respect, the LISP tutor provides the learner with a display of the current goal stack and the tutor contains a powerful LISP syntax checker, so that the learner can concentrate on conceptual problems.

After introducing these instructional principles about the learning of skills, it is important to emphasize again the connection between the procedure to be learned and the underlying model in relation to simulations, as outlined in van Joolingen and de Jong (this volume, Section 5). In principle, there are two ways of teaching operational knowledge. The first way is teaching the skill *per se*. This means learning the procedure or skill without reference to the underlying model. (In fact this is a kind of *rote learning*). The second way is to teach a procedure or skill in direct relation to the underlying model (this is a more *meaningful* way of learning).

For tasks such as LISP programming, that can be represented as production rules, no explicit relation to an underlying (conceptual) model is necessary. In this sense, tasks such as LISP programming are not really interesting for learning with computer simulations. Gott (1989) points out that adopting Anderson's approach for learning skills that are more clearly related to an underlying model (such as troubleshooting procedures) may lead to *superficial* knowledge of the skill. According to Gott (1989), one of the characteristics of experts using a skill is *adaptiveness*. This can only be acquired by relating the procedure(s) to the underlying model. Gott (1989) stresses that in performing a skill, not only conceptual and operational knowledge come into play, but also *strategic knowledge*. It can be argued of course that for many skills, knowledge of the associated underlying model will not be necessary (e.g. in learning to drive a car). This, however, probably only holds for pure psychomotor skills. Gott (1989) illustrates this by a study of Kieras and Bovair (1984). They compared two groups of learners. One group followed a rote learning technique in operating a simple control panel device, the other group received instruction that also covered the underlying model. The latter group "...learned the procedures faster, executed them more quickly, and retained them even better after a week's delay." (Gott, 1989, p. 125).

One of the instructional principles based upon this notion has already been mentioned: providing causal explanations. This instructional principle was based on White and Frederiksen's (1990) work on QUEST. The instructional goal of QUEST is not only to provide learners with insight into the composition of electrical devices, but also to teach them the skill of troubleshooting these electrical devices.

4.1.3. Knowledge acquisition skills

The third category of educational goals that we distinguished was knowledge acquisition skills. In our context these knowledge acquisition skills refer to *exploratory learning skills*. More specifically we look at these skills as *problem solving*, *learning by induction*, and *discovery learning* (see Goodyear et al., this volume). These skills are closely related to the self-regulatory skills and performance control strategies mentioned by Glaser and Bassok (1989).

Even when these knowledge acquisition skills are not an explicit goal of instruction, instructional strategies that support these skills are relevant, because these skills are, in the context of learning with simulations, the vehicle for knowledge acquisition[11].

[11]The distinction between instructional strategies and interface in relation to knowledge acquisition skills is a vague distinction. When the support contains explicit directions (e.g. the system asks the learner to make a prediction) we are tempted to call this instructional strategies, if however, the support is only instrumental (the learner is offered a scratchpad to state a hypothesis) we will see this as part of the learner interface.

The instructional principles in relation to knowledge acquisition skills, mentioned by Glaser and Bassok (1989) are:

a. *Strategies for monitoring comprehension*
When involved in a complex task, such as problem solving, several strategies can be used to monitor understanding while learning. Among these are: *asking questions* about the content, and *summarizing* the content thus far.

b. *The teacher as model and coach*
A general feeling is that knowledge acquisition skills are better transferred through *showing* the learner the target behaviour (the tutor as a model) and by correcting the learner while s/he performs the behaviour on his/her own (the tutor as a coach).

c. *Shared responsibility for the task*
Knowledge acquisition skills seem to be better acquired when working in groups. In groups, learners are asked for explanations, individual thinking is monitored by the audience and complex tasks are more manageable.
This instructional principle is closely related to the idea of scenarios in simulations, where apart from the learner different roles can be identified. These role can be played by human beings, but also be simulated by the computer.
This idea of a simulated learning companion is discussed by Chan and Baskin (1988). They distinguish a new class of ITS: Learning Companion Systems. In these kind of systems the learner does not only learn from the computer companion, but also by explaining his/her own thoughts and reasoning processes to the companion.

Additionally we mention an instructional principle that Glaser and Bassok position somewhere else:

d. *Teaching and supporting multiple learning strategies*
Glaser and Bassok (1989) take the instructional principle of teaching and supporting multiple learning strategies from (again) the work of White and Frederiksen (1987a,b; 1989; 1990). Here learners can choose, for example, to follow open ended exploration or to hear an explanation.

An important aspect of learning knowledge acquisition skills is their relation to the more domain related conceptual and operational knowledge. In general it is assumed that learning a knowledge acquisition skill is most efficiently acquired within a specific domain. Also, executing a knowledge acquisition skill (and associated regulation processes) can be frustrated by a lack of domain knowledge, as was concluded in a review by Alexander and Judy (1988). In relation to simulations this result is confirmed in a study by Njoo and de Jong (1991a).

4.2. *Instructional Design theories.*

Instructional design theories (ID theories) are theories with a pragmatic character that are "... concerned primarily with prescribing optimal methods of instruction to bring about desired changes in student knowledge and skills" (Reigeluth, 1983, p. 4). ID theories are thus *prescriptive* and *pragmatic* and, not surprisingly, show an *eclectic* character. According to Hannafin and Hieber (1989) the two basic theoretical underpinnings for ID theories spring from behaviourism and cognitive theory. Instructional design based on behaviourism "is characterized by small, complete lesson units, controlled lesson sequences, and discrete, discernible steps. Such designs typically dictate learning paths directed toward defined objectives and measurable performance criteria" (Hannafin & Hieber, 1989, p. 92). These ID theories place very little control in the hands of the learner and as such are less appropriate for learning with computer-based simulations in which learner activity is paramount (see de Jong, this volume). We will therefore concentrate on those aspects of ID theories that have a relation to cognitive theory (as was presented in the preceding section). We will discuss the Gagné and Briggs' theory (Gagné & Briggs, 1979), the Component Display Theory (Merrill, 1983), the Inquiry Teaching Theory (Collins & Stevens, 1983), and the Elaboration theory (Reigeluth & Stein, 1983).

Discussing these theories fully, would go far

beyond the scope of this article. Therefore, we will emphasize those aspects of the theories that are possibly related to learning in exploratory learning environments. This will need some reinterpretation of the theories because, as Hannafin and Hieber (1989) claim, relatively few attempts have been made to link cognitive based ID theory to computer-based learning. It is clear, therefore, that a relation between ID theory and computer-based simulations for instruction is even rarer. Notable exceptions to this are described in Merrill (1987), and Reigeluth and Schwartz (1989).

4.2.1. Basic issues in Instructional Design theories

Basically, ID theories are composed of a number (not necessarily all) of the following elements:

1. A classification of learning goals;
2. A prescription of how to break down learning goals into subgoals;
3. A description of specific instructional actions, and a prescription relating specific actions to specific (sub)goals;
4. A prescription of a sequence of instructional actions, thus defining an instructional strategy;
5. A set of conditions for the instructional actions and strategies (such as learner characteristics).

Having these elements in common unfortunately does not mean that ID theories share a standard language. This will have become clear from Section 2, where some of the ID theory approaches to learning goals have been discussed. Section 2 provided an integrated view on the learning goals that we will use in the following discussion, thereby allowing us to sometimes bypass the learning goals classification used in the ID theory under discussion.

4.2.2. The Gagné and Briggs approach

One of the most influential ID theories is that of Gagné and Briggs (Gagné & Briggs, 1979, also described in Aronson & Briggs, 1983). This theory finds its roots in Gagné's theory on the conditions of learning (Gagné, 1977) in which five different learning outcomes (learning goals) are distinguished (see also Section 2.2). In the context of learning with simulations we introduced our own classification of learning goals with basically two knowledge categories: conceptual knowledge and operational knowledge (Section 2.5). These categories are combined with a knowledge representation and a knowledge scope dimension, thus describing all kinds of learning outcomes. As a, slightly different, third type of learning goal we distinguished knowledge acquisition skills. In the classification by Gagné these dimensions are not separated and are somehow combined into his five types of learning outcomes. Here we will try to interpret Gagné's ideas using our own classification.

Gagné's approach states that for each type of learning goal a structure of *prerequisite knowledge* can be stated. This prerequisite knowledge forms the *internal conditions* that must be met before a learner can attain a learning goal. In other words, in order to reach a learning goal, the learner must possess the prerequisite knowledge or s/he must acquire this prerequisite knowledge. A second necessity for attaining a learning goal is what Gagné and Briggs call the *external conditions* which means the instructional environment. Gagné and Briggs' claim that both the internal and the external conditions differ for each type of learning goal.

Instructional design starts by creating a *learning hierarchy* for the intended learning outcome. For a procedure (in Gagné's terms *intellectual or motor skill*) this means splitting up the top level procedure into constituent subprocedures or part skills, resulting in a hierarchy[12]. Teaching follows a bottom up approach through the hierarchy, starting at the competency level of the learner.

Gagné and Briggs (1979) incorporate the mastering or refreshing of prerequisite knowledge into a sequence of phases (instructional events) in an instructional strategy. In each of the phases a number of instructional actions is given, specified for

[12]It is surprising to see that constructing a learning hierarchy is, according to Aronson and Briggs (1983), only applicable to the learning goal of intellectual skills. It is, however, not difficult to imagine similar structures for the other types of learning goals.

the different types of learning goals. The strategy consists of the following phases (following Aronson & Briggs, 1983, pp. 91-92):

1. *Gaining attention*
   This initial phase is just meant to get the learner set for the task.
2. *Informing the learner of the objective*
   For procedures this would mean showing the learner the terminal skill or providing examples and a description of the final procedure. For conceptual knowledge the learner is told what s/he should be able to do with the knowledge s/he has gained. Gagné and Briggs are less clear about informing the learner of learning goals of the knowledge acquisition type.
3. *Stimulating recall of prerequisite learning*
   As will be clear, this phase is a crucial one in the Gagné and Briggs' theory. Prerequisite knowledge is seen as essential for gaining new knowledge that is built upon it. The theory states that most of the time it is sufficient to remind learners of the prerequisites. However, we expect that, quite frequently, prerequisites have to be taught separately. The prerequisites of procedures and skills consist of subprocedures or part skills. The prerequisites of (knowledge of) principles are the related ideas in the prior knowledge of the learner. This gives the learner the opportunity to construct a well organized body of knowledge (see the ideas on schemata and organization of knowledge in Section 4.1). For knowledge acquisition skills the prerequisites are (probably!, Aronson & Briggs, 1983, p. 89) intellectual procedures.
4. *Presenting the stimulus material*
   In this phase the new information is presented. In our context this means 'presenting' the learner with the domain model by enabling interaction with the simulation. In principle there are two ways of showing the domain model: *overt* (the model is presented in one or another way) or *covert* (the learner has to induce model properties from input/output relations).
5. *Providing learner guidance*
   Learner guidance is provided in order to help the learner master the terminal objective. It may consist of giving cues for combining subprocedures or part skills or providing links to related conceptual knowledge.
6. *Eliciting the performance*
   The learner is asked to demonstrate that s/he has mastered the learning objective.
7. *Providing feedback*
   The learner is informed about the correctness of his/her performance. This should be given as precisely as possible.
8. *Assessing performance*
   This phase is in fact a recapitulation of Phase 6.
9. *Enhancing retention and transfer*
   The last phase in this instructional sequence tries to encourage transfer of knowledge by providing novel situations in which a procedure can be used, or to link conceptual knowledge to other fields of information. Of course this phase is only valid when the learning goal involves a degree of domain independence (see the Section 2.5 discussion of knowledge scope).

An important feature of Gagné and Briggs' theory for learning with computer simulations is its emphasis on the importance of prerequisite knowledge (see also Njoo & de Jong, 1991a). Gagné and Briggs' also show, that the interaction with a computer simulation is only a *part of* a comprehensive instructional strategy.

4.2.3. Component Display Theory

Component Display Theory is described by Merrill (1983). Merrill (1987) extended this theory to include learning with computer environments (such as simulations) and renamed it the 'new' Component Design Theory. We will first discuss the basic issues in Component Display Theory (following Merrill's notation we will use 'cdt' to indicate this theory) and then consider the new Component Design Theory (following Merrill, indicated as 'CDT').

Component Display Theory (Merrill, 1983) takes as its starting point a two dimensional view on learning goals that is to some extent compatible with the 'cube' idea presented in Section 2.5. It combines a content dimension (in which Merrill mentions procedures and principles) with a 'performance' dimension. The performance dimension of cdt has

two extremes 'remember' and 'use'. 'Use' is clearly related to our concept of 'compiledness', and Merrill's concept of 'remember' can be placed within our idea of 'declarative knowledge'. However, declarative knowledge as we use it, comprises more than the 'remember' concept of Merrill. The dimension of learning goal scope (domain-specific vs. generic knowledge) is not explicitly treated by Merrill.

Component Display Theory does not provide a general sequencing of the instructional actions into phases, as was done by Gagné and Briggs (1979), but concentrates on prescribing instructional actions at the micro-level and relating these instructional actions to learning goals. In the new Component Design Theory a strategy component has been added.

Merrill (1983) distinguishes 4 types of instructional actions (or 'presentation forms' as he calls them):

1. *Primary presentation forms*
   Primary presentations present information either as a generality or as an instance (both are bound to a domain, although this is less clear for general knowledge; e.g. the instantiation of a general concept is an example). As a second dimension, primary presentations use an expository or inquisitory mode. In the expository mode information is merely presented to the learner (compare with what we have called the *overt* method of presentation), whereas in the inquisitory mode the learner is asked to infer generalities from instances (compare with the *covert* way of presenting models when using simulations; see also de Hoog et al., this volume, Section 5.1). Instructional strategies can be described as a sequence of expository and/or inquisitory presentations.
2. *Secondary presentation forms*
   Secondary presentation forms are not a necessity in instruction according to Merrill, but they form the 'cream on the pudding' of the primary presentation forms. They are ".... used to facilitate the students' processing of the information or to provide items of interest, such as contextual background" (Merrill, 1983, p. 308). This is why they are denoted as *elaborations*.

   In the expository form Merrill distinguishes elaborations such as: presenting prerequisite information, presenting contextual information (who discovered this), mnemonic elaborations (helping the learner to remember the information), and presenting alternate representations of the information (such as a diagram, a chart, a formula, or other words).

   In the inquisitory form the most important elaboration is feedback, informing the learner about the validity of his knowledge. Other elaborations are 'preresponse mathemagenic elaborations'. These are prompts and hints that help the learner in stating his/her ideas.
3. *Process displays*
   "A process display consists of instructions or directions presented to the student suggesting how he or she should consciously process the information that is presented." (Merrill, 1983, p. 310). An example is to tell the student to try to form a picture of the information in an "odd or unusual way", in order to foster remembering.
4. *Procedural displays*
   Procedural displays are directions to the student about how to operate equipment in the instructional environment, such as directions to turn a page. In fact these displays have a somewhat strange position here, since they are not directly aimed at processing information.

The discussion of cdt presented so far only listed the elements of an instructional strategy (goals and educational actions). The most interesting point of cdt is of course when learning goals (content and performance dimensions combined) and presentation forms are related to each other, in other words where the prescriptive element comes in.

For the primary presentation forms Merrill (1983) distinguished two main modes of presentation: *expository* and *inquisitory*. For the expository mode different presentation forms are presented for procedures and principles both for information in general form and as an instance. As an example, general principles are presented to learners by presenting "... the name of the principle, some identification of the component concepts that comprise the principle, and some statement of the

causal relationship" (Merrill, 1983, p. 315). The presentation (exposition) of an instantiated principle, on the other hand, is according to Merrill (1983, p. 317) an *explanation* of an event within a specific domain. It would include the name of the principle, the problem situation (including conditions and descriptions that describe the event), a description of the time sequence in an event (in which the presented principle operates) and the description of causal relationships involved. Of more interest in the context of simulation environments are Merrill's prescriptions in the inquisitory mode.

Inquisitory presentation of instances of procedures involves presenting the student with the goal, the name, the materials and the equipment for performing the procedure and asking the learner to perform the steps of the procedure. Learners' performance can be assessed in two ways, by assessing the steps in the procedure and by assessing the outcome (Merrill, 1983, p. 318). In teaching general procedures the learner can be presented with a goal and asked to outline the procedure, or s/he can be asked to indicate the steps in the procedure (e.g. in the form of a flow chart). However, this only involves declarative knowledge (in Merrill's restricted sense of 'remember') of a general procedure. For compiled knowledge of a general procedure, the learner is asked to demonstrate the procedure, but this is only possible within a specific domain. In this demonstration level, simulations come into play. Teaching principles in the inquisitory mode, means presenting the learner a problem presentation and asking for the underlying principle. They may be asked to predict an outcome or to present a solution to a problem. This holds both for instantiated as well as general principles.

What we learn from this discussion is that Merrill recognizes simulations as a vehicle for teaching principles and procedures (or conceptual and operational knowledge, in our terms). It also shows that there are additional (explanatory) ways to teach those same principles and procedures. Efficient education might use a sequence of both expository and inquisitory presentations as an instructional strategy. According to Reigeluth (1983) one of the strong characteristics of cdt is that it provides a *language* for describing instructional strategies. And indeed cdt offers a quite complicated and difficult notation system. However, cdt itself does not give any prescription of such kinds of sequences. How to embed simulations into an instructional strategy remains an open question within cdt. There is, however, something to say about enhancing simulations (as a primary 'inquisitory' presentation form) with secondary presentation forms.

Feedback is the most important *secondary presentation* for inquisitory presentation forms. Feedback is, according to Merrill, most effective for learning to remember knowledge (be it procedures or principles). For using knowledge, feedback should contain (for example) information on the functioning of a procedure in a new situation, or on the adequacy of a principle in a new situation (after the learner has made a prediction).

Elaboration is another secondary presentation form. Mnemonic elaborations can be added to help the learner remember the information. For more compiled and general knowledge, elaborations accompanying inquisitory presentations (such as simulations) may consist of presenting prerequisite information, as well as presenting alternate forms of presentations (graphs, diagrams etc.). Merrill argues that more secondary presentation of all forms is needed when the presented information becomes more complex.

A final issue from cdt of relevance to simulations concerns *learner control*. Material designed according to cdt contains quite a lot of information in a number of presentation forms. Not all learners may need all information. Merrill (1983) recommends that learners are allowed to exert control not only over the *content* that they will study, but also over the *presentation form*. However, he also warns that merely offering control will not be enough, certainly not for weaker learners. They need some guidance in executing efficient control.

Merrill (1987) states that his new Component Design Theory (CDT) was developed to give an instructional design answer to developments in hard- and software and their invasion in education[13]. Merrill now makes a distinction between two 'modes' of instruction: the *tutorial model* and the *experiential model*. In the tutorial model, information (structurally represented) is

presented to the student, whereas in the experiential model the learner may directly interact with the subject matter (that is experientially represented). It is clear that learning with computer simulations as defined by us falls within the second, experiential, category. The *presentation forms* from the old cdt are replaced by *presentation functions*, that are instantiated through *instructional transactions*. Instructional transactions form the interface between the learner and the learning material. Structural (tutorial) instructional transactions encompass: *explanation, contrasting* (both expository), *naming, classifying* (both inquisitory), and *conversation* (which can be either expository and inquisitory). Experiential instructional transactions include: *demonstration* (an expository function), *exploration* (which according to Merrill can be both expository or inquisitory), *computation, assembly, operation, designing,* and *predicting* (all inquisitory).

In the old cdt, simulations were bound to the inquisitory mode of presentation, in the new CDT simulations are now tied to the experiential representation of subject matter, thereby shifting from a classification that involved *learner activity*, to a classification that involves the representation of the information. Although old cdt and new CDT contain quite a few delicate distinctions[14], we think this to be the main shift when it comes to simulations.

The main conclusion we can draw from old cdt and new CDT is that experiential representations (such as simulations) are just one form of *representation* of information. The same information can also be represented in a more structural form. Also the *presentation* of experiential information may vary, from expository to inquisitory. Moreover, CDT offers some auxiliary actions to augment simulations, but, as is true for most ID theories, the recommendations have a quite unspecific character with much uncertainty included.

[13]We will discuss here only the changes that are most relevant for learning with simulations. New CDT also adds an instructional strategy aspect to the old cdt.

[14]For example, the concept of *expository* presentation or function seems to involve more initiative from the student in the new CDT than in the old cdt.

4.2.4. Inquiry Teaching Theory

Inquiry Teaching Theory (ITT, Collins & Stevens, 1983) is of all ID theories discussed in this section the most empirically based theory. Collins and Stevens designed their theory on the basis of observations of teaching behaviour in a number of domains. The theory is of particular importance for learning with computer simulations because it is mainly concerned with *discovery learning*. The theory, therefore, is not concerned with learning goals such as learning facts or concepts, but stresses the learning of principles and theories (causal structures as Collins and Stevens call them) and the derivation of these theories (comparable to our knowledge acquisition goals).

Apart from these top level learning goals the ITT contains a number of subgoals, 10 instructional actions, and guidelines for selecting and sequencing these actions (thus providing instructional strategies).

For the instructional goal of teaching a general principle or theory, Collins and Stevens distinguish the subgoals of debugging incorrect hypotheses and teaching how to make predictions. For deriving principles or theories the (instructional) subgoals are: teaching what questions to ask, teaching the nature of the theory, teaching how to test a principle or theory and teaching how to verbalize and defend a theory (Collins & Stevens, 1983, p. 258). Subsequently Collins and Stevens (quite independently from the learning goals and subgoals) describe 10 instructional actions, of which we will highlight a few:

- *Varying cases systematically*
  Presenting specific situations to learners should occur in a systematic order. As an example the teacher varies the value of only one factor, keeping the others constant and shows the change in value of the dependent variable. The teacher may also show the range of variability of specific factors when other factors are kept constant. These kinds of variations will provide the learner with insight into the model relations. Collins and Stevens place the control over these variations in the hands of the tutor. In the case of simulations, the learner can be guided to perform these kinds of systematic variations.

- *Forming hypotheses*
  Students are prompted to state general rules about a domain. Collins and Stevens (1982) list a number of these kind of *'prodding'* techniques.
- *Evaluating hypotheses*
  Here learners are guided to make systematic variations in order to test their hypotheses. The instructional action is therefore analogous to the first one mentioned: varying cases systematically.
- *Considering alternative predictions*
  After the learner has made a prediction, the tutor proposes alternative predictions and asks the learner why these should be poorer or better predictions than s/he has made him/herself.
- *Entrapping students*
  After the learner has stated a rule, the tutor proposes a factual situation that questions the rule the learner has stated.
- *Tracing consequences to a contradiction*
  After the learner has stated a rule that has a misconception, the tutor asks the learner to make deductions that s/he will not agree with. This approach comes very close to a Socratic dialogue.

All of these instructional actions are clearly applicable to learning with computer simulations. However, Collins and Stevens use in their discussion the concept of *cases*, which means instantiations of more general models. For example a model on climate and agriculture might contain cases on Louisiana (rice will grow) and on Washington (rice won't grow). Introducing cases in a simulation can be regarded as an instructional measure in itself.

Collins and Stevens (1983) present what they call a *dialogue control structure*, in this way providing an instructional strategy. The instructional strategy is quite simple and is composed of four parts. First, the tutor selects cases according to the rules:

- present cases that illustrate more important factors before cases illustrating less important factors;
- present concrete before abstract factors;
- present more frequent or important cases before less frequent cases.

In the second part the tutor questions a learner on the model relations for a specific case thereby identifying bugs and misconceptions in the learner's knowledge. These bugs and misconception are placed on the tutor's *agenda* and instructional actions are applied to them. The order in which bugs are treated is set by a number of *priority rules* such as 'errors before omissions'.

By putting much stress on discovery learning Collins and Stevens' Inquiry Teaching Strategy is of direct relevance to learning with computer simulations. The general idea of the theory is to introduce *cases* and to cure the learner's bugs and misconceptions by means of a number of instructional actions. However, ITT does not state relations between types of bugs or omissions and specific instructional actions.

4.2.5. Elaboration theory

The Elaboration Theory of Instruction is described by Reigeluth and Stein (1983). The theory is mainly concerned with describing instructional strategies (the macro level, as Reigeluth & Stein call it). For the teaching of smaller units of knowledge they refer to Merrill's (1983) Component Display Theory.

The Elaboration theory distinguishes between three types of content that can be regarded as learning goals: concepts, principles and procedures. The last two resemble the conceptual and operational knowledge as we have described them in Section 2.3.

The Elaboration theory criticizes learning hierarchy approaches (such as Gagné and Briggs' theory) for denying the importance of general views on a principle or procedure. Using a bottom up learning hierarchy approach may result in fragmented and not related knowledge. However, Elaboration theory recognizes the importance of the learning prerequisites in a hierarchy, but states that this approach is only satisfactory for small parts of a course. The general teaching approach taken in a course should proceed from a small number of simple and fundamental ideas from a theory or procedure and then proceed by presenting more complex information (what Reigeluth & Stein call a

simple-to-complex or *zoom lens* approach). Elaboration theory tries to integrate this general approach with the learning prerequisites approach.

The instructional strategy according to Elaboration theory starts by presenting the learner with what Reigeluth and Stein (1983, p. 343) call an *epitome*. Epitomes are simple but fundamental ideas from the content to be taught that represent the gist of the target knowledge. Information in epitomes is presented at an *application* level. Epitomes differ from summaries in that summaries do not present just a small number of central ideas and are presented at a more abstract level[15].

These epitomes and the subsequent zooming in or simple-to-complex instructional process, are different for the different types of learning goals (concepts, principles, procedures).

For *procedures*, simple-to-complex means that the shortest procedure is presented at the start of the instruction. Complexity is added by presenting alternative paths, in this way also enlarging the application area of the procedure. For *concepts* and *principles*, the starting information is some central, fundamental ideas from the domain presented at an *application* level. This means that learners could apply the information, it is not *abstract* information. Instruction proceeds by presenting the remaining, less central, concepts and principles. Reigeluth and Stein (1983) present some organizational principles for concepts, theories and procedures that may guide the simple-to-complex sequencing.

Elaboration theory recognizes the importance of learning prerequisites. Learning procedures, concepts or principles at a certain level of complexity may involve learning these prerequisites (prerequisites of, for example, a particular principle are concepts and change relationships). Reigeluth and Stein label these prerequisites *supporting knowledge*. This supporting knowledge is presented together with the epitomes.

We started by saying that Elaboration theory was developed in order to overcome the overall fragmentary approach of content typical of learning hierarchy approaches. One of the ways to overcome this is the simple-to complex sequencing. However, after zooming in, the learner has to return to a higher level of knowledge and zoom in again on a new piece of knowledge, in order to cover the complete target domain. In order to integrate the knowledge acquired in this way, elaboration theory offers the learner *summarizers* and *synthesizers*. Summarizers are aimed at memorizing knowledge (so, in our terms, at the *declarative form* of knowledge), whereas synthesizers are aimed at deeper knowledge (our *compiled form* of knowledge). Additionally Elaboration theory offers *analogies* that relate new ideas to existing ideas.

A final important aspect of Elaboration theory is that it allows for *learner control*. First the learner is allowed to choose his/her way through content as it is presented in the simple-to-complex hierarchy. Presenting epitomes at different levels makes the learner informed, in this way supporting the learner in making right decisions. Second, the learner may be allowed to choose a summary, synthesizer, or analogy.

Relating the Elaboration theory to learning with computer simulations leads to the conclusion that it might be fruitful in teaching complex conceptual models or procedures to start with more central models and procedures and gradually move to the more complex models and procedures. Elaboration theory offers support for such a design.

4.2.6. Conclusions from Instructional Design theories

Having discussed four of the most prominent ID theories we can conclude that each of them offers (different and complementary) guidelines for embedding simulations in an instructional environment. Summarizing, these guidelines are:

- Interacting with a simulations is only a part of a more comprehensive instructional strategy, in which for example prerequisite knowledge is also taught. (Gagné and Briggs);
- Interaction with the simulation itself may be

[15]We realize that the information presented here just gives an impression of the Elaboration theory. Reigeluth and Stein (1983) present a number of examples that help clarify the idea of epitomes.

accompanied by informative feedback and elaborations. Moreover, information represented in a simulation can also be represented in a more structural or static way. The presentation of information can be either expository or inquisitory, and these two forms of presentation can be alternated. (Merrill)

- Learners can be provoked to perform specific learning processes and learner activities by tutor controlled variations in the simulation, and by tutor initiated 'prodding' techniques. (Collins and Stevens)
- Complex models and procedures can be taught by starting with central and simple elements of these models and procedures and subsequently presenting more complex models and procedures. (Reigeluth and Stein)

All four of these elements can be retraced in a recent article that tries to relate ID theory to learning with computer simulations (Reigeluth & Schwartz, 1989).

Reigeluth and Schwartz (1989) describe three types of knowledge that can be learned with simulations. First, there is operational knowledge, sequences of steps and/or decisions. Second, there is conceptual knowledge in which they make the distinction between processes and cause-effect relationships. The latter distinction looks similar to our distinction between dynamic and static underlying models (Van Joolingen & de Jong, this volume, Section 3.1.2). However, Reigeluth and Schwartz restrict the dynamic models only to naturally occurring phenomena, in our view an inessential restriction. Moreover, it remains to be seen whether the distinction between dynamic and static models has an effect on the way instruction should proceed.

Reigeluth and Schwartz distinguish three main stages in the instructional process of learning with simulations:

- an *acquisition phase*, where the knowledge is presented to the learner;
- an *application phase*, in which the acquired knowledge is generalized to a larger set of situations;
- an *assessment phase*, in which the knowledge of the learner is assessed according to some criterion.

These phases are preceded by an introduction phase in which the learner is informed about the scenario of the simulation and about the goals. This sequence of phases clearly resembles Gagné and Briggs' instructional strategy (see above).

In the acquisition phase the simulation can be used in two modes: a *demonstration mode* in which no learner activity is involved and a *discovery* mode in which there is learner activity. These two modes resemble Merrill's expository and inquisitory presentation forms.

In the application phase conceptual and operational knowledge is transformed from a declarative to a compiled form. In this phase Merrill's secondary presentation forms (as feedback and elaboration) come into play. Reigeluth and Schwartz distinguish *natural* and *artificial* feedback.

For presenting the simulation Reigeluth and Schwartz identify a number of *variations*. Some of these refer to presenting prototypical examples and variations on these similar to the ones discussed by Collins and Stevens. Regarding the complexity of the simulation, Reigeluth and Schwartz recommend presenting the content as an *integral whole* when the content has only a limited number of principles or procedures, but to use a simple-to-complex sequence when the terminal knowledge is complex (see Reigeluth & Stein, 1983).

Reigeluth and Schwartz (1989) stress that their recommendations are not sustained by empirical evidence and they hope more research will be performed on the basis of their recommendations.

4.3. *Intelligent tutoring systems for exploratory learning.*

Research on intelligent tutoring has resulted in several exploration-based learning systems that also incorporate some adaptive instructional support functionality. We will first describe the instructional features of a number of simulation-based ITS's (SOPHIE, STEAMER, QUEST, MACH-III,

IMTS, and Smithtown). Then we will do the same for a number of exploration-based ITS's *not* employing simulation (GUIDON, the ACT*-based tutors, the EUROHELP/FysioDisc Coach, and SHERLOCK). The discussion of these systems will be concluded with a brief review of evidence for their effectiveness.

4.3.1. Simulation-based ITS's

All but one of the simulation-based ITS's described below involve operating and troubleshooting mechanic and/or electronic devices (operational knowledge), and the associated mental device models (conceptual knowledge). This seems to be a general emphasis in the field itself (see for instance Psotka et al., 1988), rather than a bias in our selection.

SOPHIE

One of the classic examples of simulation-based ITS is SOPHIE, a SOPHisticated Instructional Environment for teaching electronics troubleshooting (e.g. Brown et al., 1982). The basic instructional philosophy of the project is that of creating a 'safe' reactive learning environment, in which learners can "formulate, test and witness the consequences of their ideas without having to worry about possible catastrophic consequences" (Brown et al., 1982, p. 229). In the context of electronics, this is expected to improve a learner's understanding of troubleshooting strategies and tactics (operational knowledge), as well as of electronic laws, circuit causality, and functional organization of the particular device (conceptual knowledge).

Three different versions of SOPHIE have been created in the course of the project, each one improving upon its predecessor in particular ways. Since SOPHIE-II is an extension of the SOPHIE-I system, and since in contrast to SOPHIE-III the resulting instructional environment has been fully implemented, we will focus on SOPHIE-II. It consists of a simulated lab environment (SOPHIE-I), augmented with a 'canned' articulate expert troubleshooter and a gaming mode. We will first describe the functionality of each of these three ingredients, and then illustrate them with a typical SOPHIE-II course setup.

In SOPHIE's *lab workbench*, learners can troubleshoot faulty electronic circuits while being monitored and supported by the lab's coach. Functional block and circuit schematics are supplied off-screen, and, in contrast to SOPHIE's descendants, there is no graphics interface to the simulated device. The learner can request a measurement or replacement verbally ('what is the voltage at Q6?'), to which the SOPHIE system will respond in a similar way. A replacement request will only be honoured if it has been motivated by an explicit hypothesis. The automated lab coach monitors the learner's troubleshooting performance, and it can step in if a requested measurement is redundant, if an hypothesis is inconsistent with available test results, or if the learner directly asks for help.

The *articulate expert* of SOPHIE-II can provide the learner with a demonstration of optimal troubleshooting performance. After it has outlined the global approach of a lesson, the expert allows the learner to select which of the seven functional blocks of the device is to be malfunctioning. After SOPHIE itself has covertly selected the particular fault at the underlying circuit level, the expert demonstrates its troubleshooting approach, explaining things along the way. Each time the expert has decided upon a particular measurement, the learner is asked to predict whether the expected value will be normal, too high, or too low (based upon knowledge of the selected malfunctioning block). If the learner's qualitative prediction is wrong, the expert invokes the SOPHIE lab to obtain the quantitative measurement. If the learner is still unable to restate the measurement in qualitative terms, the expert will do so, after which the demo continues. As soon as the expert has found the faulty block, the learner can enter SOPHIE lab in order to locate the faulty component within the block's circuit for him- or herself.

In addition, SOPHIE-II comes equipped with a very interesting *gaming mode*, in which an 'inserter' party introduces a particular fault, which a 'debugger' party must then locate in the most efficient way. The scores of both parties are not only related to the quality of measurements being taken by the 'debugger', but also by the quality of predictions made by the 'inserter' for each intended measurement. This game is intended to promote a cost-effective approach to troubleshooting, and in addi-

tion provides for a highly motivating application of a player's causal and teleological understanding of the device.

While active engagement of the learner is emphasized in the gaming mode, it is also stimulated by the other SOPHIE-II ingredients. The expert demo, for instance, nicely illustrates that demonstration need not allow a learner to 'sit back and relax', but can also require active participation. More generally, the combination of expert demonstration and relatively unconstrained troubleshooting is a deliberate pedagogical choice made by the SOPHIE team: while expert demo's can make sense of apparent conflicts a learner may (or has) encounter(ed) in the lab, personal lab experiences in turn provide for a more motivated and purposeful observation of expert behaviour.

Brown et al. (1982) illustrate the above functionality with a typical 12 hour SOPHIE mini-course, which they have used for a variety of experiments. The first of four sessions in this course introduced learners to the SOPHIE system itself, reviewed some basic electronics of regulated power supplies, described the particular power supply at hand, and ended in a question-answering period and informal discussion on troubleshooting strategies. In the second session, learners alternately watched the expert and worked on preselected faults in the SOPHIE lab themselves. The third session consisted of some individual practice on secondary faults (a component blow out by a primary fault), and of some more troubleshooting exercises in collaboration with another learner. In the final session, teams of two learners each played the SOPHIE game against one another, after which the course finished with some individual exercises.

SOPHIE-II is an early, but impressive example of the variety of instructional features surrounding a simulation, and of its use in a wider curriculum structure. Nevertheless, the project advanced to a third SOPHIE version, aiming at more suitable knowledge representation for instructional purposes. In particular, SOPHIE-III has replaced the quantitative simulation model by a qualitative one, and the 'canned' troubleshooting expert by an electronics/troubleshooting expert couple reasoning over the qualitative model. SOPHIE-III has never been fully implemented, part of the reason being that the project needed a more advanced notion of what it really means to 'understand a device'. As Wenger (1987, p. 76) states, "... the evolution of the SOPHIE project can be viewed as a search for the right ingredients of 'cognitive fidelity' in expert reasoning". The next project, STEAMER, has explored this issue in a more pictorial way.

STEAMER

The STEAMER project (e.g. Stevens & Roberts, 1983; Hollan, Hutchins & Weitzman, 1984) has extended the concept of simulation-based learning into the realm of interactive, manipulable graphics. The STEAMER system simulates a complex steam propulsion system for large ships. Learning how to operate a steam plant requires years of instruction and experience, in which engineers must acquire a large set of operational procedures for dealing with both normal and abnormal conditions. Although the ultimate learning goal involves these procedures (operational knowledge), a basic assumption of the project is that their acquisition and execution leans heavily on an accurate mental model of underlying device principles (conceptual knowledge). The design of STEAMER has therefore concentrated on how to have people learn a sophisticated mental model of a complex physical device.

The STEAMER system allows for instruction in three different modes. In *free manipulation mode* the learner can freely explore the simulated steam plant. The interactive graphics interface to the quantitative simulation model has been designed to as much as possible reflect the mental device models of a steam plant expert ('conceptual fidelity'). About 100 different steam plant views are available to the learner, intended to reflect the expert's abstraction hierarchy of qualitative device models. In exploring the steam plant, a learner can freely move up and down this hierarchy, interacting with increasingly abstract or specific representations of the system or its components. Each representation provides the usual information on (sub)system state, and the usual handles available for plant control. However, understanding of the device is facilitated by the addition of causal topology information (e.g. depicted flow through a pipe) and simulation control handles which would not be available in real-life. Apart from this feature, no further instructional support is available in this

mode, and learners have full control over exploration.

Operating a steam plant requires knowledge of a large number of procedures, laid down in an extensive set of manuals. In order to help learning about these procedures, STEAMER also has a *tutorial mode* in which procedure execution can be practised under surveillance. The learner works through a procedure by sequentially selecting the next step from a randomly ordered menu of steps, after which its effects on the system can be observed. When the learner selects a wrong step, he or she receives feedback in terms of the underlying engineering principles that argue against the selected step.

Typical for the STEAMER project, the approach to feedback and explanation is an abstract one: procedures are characterized in terms of generic device components and related engineering principles. For instance, a particular procedure for starting the main condensate pump is rationalized in a relatively device-independent way: "According to the principles which require that whenever you admit steam into a closed chamber you should first align the drains, before opening valve 13 you should align the drain valves FWD-E254 and FWD-E239." (Stevens & Roberts, 1983, p. 19). This shows that STEAMER does not merely address knowledge about a specific device ('domain'), and that the acquisition of relatively generic engineering knowledge is in fact a major issue here.

STEAMER actually comes with a third instructional mode, called the *feedback minilab* (Forbus, 1984). This lab, much like SOPHIE's workbench, allows the learner to explore the detailed functionality of low-level control components. Learners can assemble a device from generic control components, such as a comparator, and subsequently do some testing on it. Although the lab has some limited coaching support, e.g. based on the recognition of common bugs, its main instructional significance lies in helping learners to understand about abstract component functionality.

The common factor in each of these STEAMER modes is that they all aim at conceptual fidelity in terms of *abstraction levels*. The free manipulation mode supports abstraction in terms of a hierarchy of qualitative device representations. The tutorial mode describes and explains a specific procedure as an instantiation of a more generic one, defined in terms of generic objects and engineering principles. And the feedback minilab in turn supports the understanding of abstract object functionality. As such, the STEAMER project has continued SOPHIE's search for the appropriate way to represent domain knowledge towards the learner. According to Gott (1989, p. 39), however, a major shortcoming of the project is that, in its emphasis on device knowledge, it has neglected expert knowledge of troubleshooting procedures: "...there is no simulated cognitive model of procedural performance to support procedural instruction the way the device simulation supports the mental model instruction". We will see that subsequent projects (especially MACH-III and IMTS) have tried to improve upon this state of affairs.

QUEST

The QUEST system (Qualitative Understanding of Electrical System Troubleshooting; e.g. White & Frederiksen, 1989; 1990) teaches troubleshooting of simple electronic circuits, which consist of a couple of switches, resistors, capacitors, etc. The overall project is not especially interested in teaching electronics troubleshooting as such. Rather, the central question is how to have people learn a sequence of causal models (of particular domain phenomena) by means of qualitative simulation. In its focus on qualitative mental models the QUEST enterprise closely resembles the SOPHIE and STEAMER projects. Unlike these two, however, there is a strong emphasis on learning carefully sequenced, *progressively complex* domain models, each of which is built on top of the previous one.

Since its instructional features are very much tied in with cognitive theory, QUEST has already received a fairly detailed discussion in Section 4.1. We will therefore just mention a major assumption underlying the QUEST work which has major implications for the instructional environment. The assumption is that, as long as one appropriately designs the sequence of models and the accompanying problem sets to which the learner is exposed, the learner will develop hardly any misconceptions at all. In line with this, the QUEST system has not been equipped with any facilities for misconception diagnosis and remediation at all, which sets it apart

from many other ITS's.

MACH-III

The MACH-III system (e.g. Kurland & Tenney, 1988; Kurland, 1989) deals with maintenance and troubleshooting of a complex radar device, and is currently being used as a trainer in a real curriculum. Its instructional features have been derived from an extensive analysis of the cognitive task at hand, and the major problems that novice engineers have with it. Although this task analysis has been done for radar troubleshooting only, the problems revealed most likely pertain to complex systems maintenance in general (see for example Nawrocki, 1987).

The necessary information for troubleshooting this particular radar device is distributed over a large set of wiring schematics, and a large set of Fault Isolation Procedures (FIP's). However, critical information needed for a problem is often too difficult for a novice to understand, and is often hidden in, or missing from the standard. Fault Isolation Procedures direct the mechanic through a branching sequence of tests, without providing him or her with any sense for why or where testing is done, or what has been ruled out. Furthermore, novices lack adequate (e.g. functionally organized) device models, as well as knowledge of higher-level troubleshooting strategies.

In order to address these problems, five major instructional principles have guided the design of MACH-III[16]:

1. *Show the problem space.* Research on expert problem solving has shown that diagrams can be very useful, both as an extension of working memory and as a way of representing information difficult to describe in propositional form (such as feedback loops). According to Kurland and Tenney, appropriate diagrams should help a novice by providing conceptual support, temporary models, and higher-level organizers for new information.
2. *Practice troubleshooting on increasingly difficult problems.* Troubleshooting exercises should use increasingly complex representations of the device, ranging from high-level flows of information and power to very low-level circuit schematics. This notion is very much in line with QUEST's emphasis on careful sequencing of progressively complex models to be learned.
3. *Provide a range of procedural help.* This principle involves fairly direct support that should be available within a troubleshooting exercise, such as helping the learner to locate a particular component, intervening when an incorrect switch setting invalidates a particular test, and providing part replacement information.
4. *Provide supports to make reasoning strategies explicit.* According to the designers, this is where MACH-III is expected to contribute most, and the principle can be implemented in several different features:
   a. The learner should select from a menu his/her most likely fault hypothesis (a specific impaired function, e.g. 'Doesn't remove feedthrough noise'), before testing any of the components involved in that function. This induces a predictive attitude in the learner, and guides him or her away from vague or incorrect fault hypotheses, or from associating components with the wrong function. It also provides a handle for learner modelling.
   b. The learner should be alerted to test results which are inconsistent with the selected fault hypothesis.
   c. Correctly functioning circuit portions should be visually different from still suspected areas, thus reducing the search space of possible faults.
   d. A window which shows the learners previous hypotheses, tests, and results should help the learner to keep track of things, and to compare his or her own performance to expert performance.
   e. A facility which gives increasingly specific hints should prevent the learner from getting stuck without giving the solution away too easily.

[16]Since the paper from which these principles have been taken (Kurland & Tenney, 1988) describes MACH-III in its early implementation stage, it is not entirely clear to what extent the MACH-III version now in use actually incorporates them; the brief paper on the latter version (Kurland, 1989) seems to describe the full implementation of the first two, and a partial implementation of the other three principles.

f. Modelling an expert's solution path does not only show the correct sequence of steps, but can also make explicit the expert's reasoning and problem-solving strategies behind those steps.

5. *Provide explanations.* Simulation-based learning should benefit from explanations of five different types:
   a. *Device function explanations*, which answer questions such as: What is the device for? What does it do? and How does it do it?
   b. *Topology explanations*, which answer questions related to location, such as: Where does a signal come from and go to? and Where is the device?
   c. *Modularity explanations*, which address the higher-level grouping of devices and signals into larger modules.
   d. *Simulation behaviour explanations*, which tell the learner what he or she is seeing on the screen, what the simulation is doing at that moment.
   e. *General explanations*, which are based on the classification of devices according to their high-level function (such as 'signal refinement'), or on the classification of high-level troubleshooting procedures (which for instance allows for the selection of an appropriate troubleshooting strategy); the latter type of general explanation is closely related to 'meta-cognitive reasoning processes' of the expert.

The MACH-III version currently in use (Kurland, 1989) incorporates the following features (with references to the principles involved):

1. *Summary diagrams* portray multiple views of the system at different levels of detail (principle 1), much in the abstraction spirit of STEAMER. These diagrams summarize the highly detailed circuit schematics from a troubleshooting point of view by showing test points, replaceable devices, and their approximate physical arrangement. Diagrams can be purely physical, purely functional, or mixed, and each type of diagram can be interactively used for troubleshooting exercises.
2. The MACH-III system includes a *database of all possible faults*, ordered on curriculum topic and problem complexity (principle 2). Problem complexity is determined by the instructor, and involves fault dimensions such as signal type involved, number of possible measurements and replacements, etc. Within a particular category, problems can be selected at random, but an instructor can also specify which exercise subset should be presented at first.
3. *Troubleshooting trees* for each subsystem organize troubleshooting knowledge and procedures into a functional hierarchy. Each tree contains a functional grouping of devices which may be involved in particular malfunctions, and the system 'greys out' items that have been tested to show what is left (principles 1, 4c and 4d).
4. A (selectable) *troubleshooting Advisor*[17] helps the learner to decide what functional hypothesis to consider, and what next test to perform (principle 4e?).
5. A (selectable) *troubleshooting Critiquer* gives feedback on the learner's last step, according to the component selected for testing, its position in the troubleshooting tree, and the learner's previous actions (principle 4b?).
6. An *explanation system* (principle 5) relies on canned explanations of devices (purpose, context, behaviour, etc.) and troubleshooting tests (in terms of their high-level reasons).
7. Three different troubleshooting modes are supported: (a) *Magic Mode*, in which all signal flow is shown, and both accessible and inaccessible devices may be tested; (b) *Real-Life Mode*, in which no signal flow is shown, only accessible devices may be tested, and the Advisor/Critiquer functions are operational; and (c) *System Demonstrates Mode*, which is much like Real-Life Mode, but in which the expert demonstrates optimal troubleshooting, explaining things and changing the diagram along the way (principles 4f and 5e).

The above MACH-III implementation is currently being used in the training of radar mechanics. Preliminary observations show that learners make

[17] The exact functionality of both the Advisor and the Critiquer cannot be derived from the brief paper which mentions these two components (Kurland, 1989).

heavy use of the troubleshooting trees, which capture the implicit organization of Fault Isolation Procedures in an explicit functional hierarchy. Another finding is that, after a few trials in Magic Mode, students prefer to work in Real-Life Mode, hardly making use of System Demonstrates Mode. As a final observation, students only ask for advice when instructors encourage them to do so, and are not much inclined to read the Critiquer information.

IMTS

The Intelligent Maintenance Training System (e.g. Towne et al., 1988, 1990; Towne & Munro, 1988; 1989), like MACH-III, provides a training environment in which learners can practise troubleshooting a complex device. However, unlike MACH-III, IMTS is a relatively generic system, which allows for the authoring of intelligent simulation learning environments geared to troubleshooting of all kinds of hydraulic, mechanical, electrical, or mixed systems. While MACH-III's instructional features were tailored to a particular radar device, the instructional environment of IMTS is mostly derived from the simulation model at hand.

The main instructional feature is a *sequence of troubleshooting exercises*, each of which involves a particular malfunction introduced in the simulation model[18]. An exercise consists of a configured simulation, accompanied by an initial operator complaint (such as 'The override light is coming on in stand-by mode'). The learner's task is to locate the malfunctioning component in an efficient, nonredundant way. In systems of reasonable complexity, the difficulty of this task usually arises from the fact that abnormal behaviour of the faulty component propagates through the system, causing other components to misbehave as well[19]. Furthermore, the initial complaint may be vague and unhelpful, or even incorrect. Since the complaint is just a piece of 'canned' text attached to a malfunction, and the precise nature of all possible malfunctions has also been specified by the author, both sources of complexity are under control of a human instructor.

In order to have the trainee actually learn something, each successive exercise in the practice sequence should challenge his or her current level of understanding and proficiency. For deciding upon a new exercise, IMTS selects the most 'optimal' one from the set of currently remaining troubleshooting exercises. The value of each remaining problem is determined on the basis of its predetermined difficulty level, of its conceptual distance from the last problem, and of learner model dimensions (such as a metric indicating the current learner's learning speed). As such, IMTS attempts to provide the learner with an optimal sequence of problems, adapted to the dynamically changing instructional situation.

The support given *within* a particular troubleshooting exercise is determined by the quality of the learner's testing behaviour. Initially the learner is completely free in how to approach the problem at hand. He or she can test individual components by taking measurements on them, or by replacing them by 'spare' ones, after which the simulated effects can be observed. At all times, the learner can ask for additional information on a particular component. Currently, this can either be a bitmapped photograph, or some text on the component's operation or purpose[20]. If desired, IMTS can be asked for expert help in interpreting test results, in which case the system responds with something like:

> The voltage at X was Y, which is normal, so we now know that the following components are working normally: <list of subsystems>. I still suspect the following: <...>.
>
> (adapted from Towne et al., 1988, p. 33).

[18]Some 'troubleshooting exercises' actually involve *no* malfunction at all, which, according to Towne and Munro (1988), reflects a diagnostic situation frequently encountered in the field.

[19]This kind of propagation involves the abnormal behaviour of other components within their behavioural range (e.g. a flashing warning light). However, a malfunctioning component may also cause other ones to break down themselves (e.g. a blown warning light). Simulation of such 'cascading errors', as well as the simultaneous introduction of multiple faults, will be incorporated in a future IMTS version (Towne & Munro, 1988).

[20]A third option, which is still under development, will allow the learner to operate a particular component in isolation, that is, outside of the current simulation context (cf. SOPHIE's workbench and STEAMER's feedback minilab).

If the learner doesn't know how to proceed, he or she can also ask IMTS for a suggestion, such as:

> Measure the voltage at X. A value of Y would be normal. If normal, the failure is one of <...>. If abnormal, the failure is one of <...>.
>
> (adapted from Towne et al., 1988, p. 32).

In addition, one may ask IMTS for a hint on the nature of the problem, which (like the initial complaint) consists of some 'canned' text tied to the malfunction at hand.

If the learner gets lost at the early stage of exploration, the above hint is automatically displayed. However, when performance falls seriously below certain predetermined standards, such as when many redundant tests are carried out, IMTS steps in with *coaching* support. First of all, the student is asked to identify those components he or she still suspects on the basis of completed tests. If the malfunctioning component is among those suspected, IMTS guides further troubleshooting by means of expert test interpretations and suggestions. However, if the learner does not suspect the actual fault, IMTS engages him or her in a (partial) debriefing discussion about the tests already performed.

In addition to the above, several types of *between-problem* support are available. On completing an exercise, for instance, IMTS will present a (pre-authored) technical summary of the problem. After this, the learner has the opportunity to watch 'Profile', the expert component of IMTS, troubleshoot the same problem, giving rationales for testing along the way. Another option available is debriefing by means of a replay, during which Profile criticizes each of the trainee's testing actions. Alternatively, the learner may choose to replace the malfunctioning component, carry out some confirmatory tests, and re-introduce the fault again. The learner may even request the introduction of another fault in the simulation, which is particularly useful for finding out what would have happened if a wrongly suspected component was in fact the faulty one. Judging by the currently available literature on IMTS, all these types of support are entirely optional.

The instructional policy behind IMTS is a very *non-invasive* one. Since, as Towne and Munro rightly state, it is not reasonable to expect learners (or even experts) to perform optimally, "...the coach must give the learner some room to explore, thereby gaining the experience of monitoring his own performance" (Towne & Munro, 1988, p. 486). Designing the instructional environment of IMTS has been guided by a large set of instructional principles[21]:

1. Instruction should be relevant to the problem-solving context.
2. Provide immediate feedback on errors.
3. Where possible, attempt to sustain diagnostic exercises by postponing instructional content that would reveal the solution to the exercise until after the exercise is completed.
4. An intelligent training system should respond quickly to student's moves and requests for assistance.
5. A good tutor should explicitly identify the goal structure of the problem domain.
6. Minimize working memory load for students.
7. Prevent superstitious behaviour (protect students from chance positive outcomes).
8. Students should approach target skills by successive approximation.
9. Protect students from building extended chains of misconceptions.
10. Protect students from chance negative consequences from appropriate moves.
11. Be opportunistic in providing instruction as context permits.
12. Maintain the credibility of the intelligent trainer.
13. Before giving advice, be sure the issue used is one in which the student is weak.
14. When illustrating an issue, use only an example (an alternative move) in which the result or outcome of that move is dramatically superior to the student's move.

[21] Several of these principles have been derived from ACT*-based projects (e.g. Anderson et al, 1987). The principles 13-21 originated in the WEST-project (Burton & Brown, 1979), an in-depth exploration of coaching which was in turn inspired by SOPHIE research. Since some of these 21 principles, like numbers 2 and 17, are in partial conflict, not all were rigorously followed (Towne & Munro, 1988).

15. After giving the student advice, permit him or her to incorporate the issue immediately by allowing the turn to be repeated.
16. If a student is about to fail, interrupt and coach only with moves that will keep him or her from failing.
17. Do not tutor on two consecutive moves, no matter what.
18. Do not tutor before the student has a chance to discover the issue for himself or herself.
19. Do not provide criticism alone when the coach interrupts. If the student makes an exceptional move, identify why it is good and congratulate him or her.
20. Always have the computer expert play an optimal game.
21. If the student asks for help, provide several levels of hints.

The non-invasive instructional policy implies that learners normally have a large amount of control over the scenario. However, the IMTS team is working on an instructional feature which is much more constraining than the above ones (Towne & Munro, 1988). IMTS will be expanded with an authoring mode in which experts can perform a demonstration sequence of troubleshooting operations, and enter explanations of their procedures and system responses along the way. The result would be a *guided simulation* lesson, in which the learner can interact with a replay of this session, following appropriate instructions on what to do next. This constraint on learner activity would, according to the authors, be particularly useful for what they call 'procedure drills' (compiled operational knowledge).

Another instructional feature under development pertains to the level of abstraction at which the simulated system is displayed to the learner. A particular IMTS simulation revolves around a 'deep model' of the device at hand, which has been authored by means of a generic object library. The learner interacts with 'simulation scenes', whose appearance is separately authored, but whose behaviour is governed by the simulation model. However, apart from an increased fidelity of individual objects (e.g. a switch-like icon instead of an electrical symbol), such scenes remain very close to the model as designed for driving the simulation. This may not be the best design for easy understanding of the simulated system. The project is therefore working on an authoring tool which allows for the composition of *pedagogical views*, displaying (parts of) the system in an instructionally useful way. The same issue was very much emphasised in the STEAMER project, which is related to the QUEST-like concept of progressive implementation, and also featured as summary diagrams in MACH-III.

Smithtown

Smithtown (Shute, Glaser, & Raghavan, 1989; Shute & Glaser, 1990) is an intelligent 'discovery world' in which learners can explore the laws of basic micro-economics by manipulating a set of simulated economic markets. The system primarily assists the learner in becoming more systematic and scientific in the discovery of laws for *any* given domain. It thus aims more at the development of 'scientific inquiry skills' (knowledge acquisition knowledge; see Section 2.5.2) than at the acquisition of domain-specific conceptual knowledge.

Instead of setting up a curriculum (e.g. a sequence of problems) for the learner, Smithtown offers complete freedom to explore all aspects of the simulated markets. Learners can manipulate variables (e.g. goods, labour costs, population income), observe the effects (e.g. on prices), and organize the information gathered in an effective way. Data organization is supported by a number of online tools: a notebook for collecting data, a table to organize data from the notebook, a graph utility to plot data, and a hypothesis menu to formulate relationships among variables. In addition, learners can inspect the history of actions, of data, and of concepts learned.

A typical episode of Smithtown exploration begins when the learner selects a market (that is, one type of goods), and indicates the variables which will be under focus in the experiment to come. The latter feature supports the learner in planful behaviour, and also informs the system of his or her experimental intentions. Next, the learner is asked whether he or she wishes to make a prediction for the experiment. If not, the learner can immediately proceed by choosing from a menu of 'Things To Do'. If the learner does want to make a prediction,

it can be written down in a dedicated window, after which the same menu appears. By offering this choice the system allows for the fact that experimentation can be more or less hypothesis-driven, depending on for example the state of exploration or the type of learner using the system[22].

The 'Things To Do' menu makes all further options accessible to the learner. Learners can request information on the current state of the market, have the system adjust the goods price in a way that moves the market closer to equilibrium, or do so themselves. They can select variables for entry in the notebook window, and display variables of particular interest in a table or a graph. Several options allow the learner to set up and run a new experiment (or experimental episode) in relation to a previous one, e.g. through 'change good, same variable(s)'.

The most interesting option on the menu, however, is to make a hypothesis. After selecting this option, learners can specify their current hypothesis by means of a submenu of *connectors* (if, then, as, etc.), a submenu of *objects* (price, surplus, demand, etc.), a submenu of *verbs* (increases, equals, has no relation to, etc.), and a submenu of so-called *direct objects* (over time, zero, along the S-curve, etc.). The learner can assemble a hypothesis (e.g. 'as price increases, quantity demanded decreases') by picking elements from several of these menus. The system can in turn parse such assembled hypotheses, compare them to its domain-specific knowledge base, and give appropriate domain-specific feedback.

Apart from evaluating hypotheses, Smithtown also monitors a learner's exploratory behaviour. When a learner is floundering or repeatedly demonstrates (predefined) buggy behaviours the system's coach will intervene. On detecting haphazard exploration, Smithtown may for example step in with the following advice (Shute & Glaser, 1990, p. 55):

> I see that you're changing several variables at the same time. A better strategy would be to enter a market, see what the data look like before any variables have been changed, then just change one variable while holding all the others constant.

As one can see, behavioural coaching addresses the learner's inquiry skills rather than his or her understanding of the economics domain itself.

Experimental evaluation revealed that about 5 hours of Smithtown exploration results in a similar degree of micro-economics understanding as approximately 11 hours of traditional classroom instruction. Thus, in terms of the domain, learning with the system seems to be about twice as effective as learning in the classroom. Smithtown's advantage is all the more interesting as its tutoring primarily addresses the subject of scientific inquiry rather than of micro-economics. In addition, a combination of classroom and Smithtown-based learning may turn out to be even more effective.

An in-depth study of individual differences between subjects who had worked with Smithtown revealed that, compared to unsuccessful learners, successful ones (1) operated in a more hypothesis-driven way, (2) explored a particular context in a more structured and in-depth way, (3) planned their experiments and manipulations to a larger extent, (4) used more powerful inquiry heuristics like 'gather sufficient data before drawing any conclusions', and (5) paid more attention to data -- and thus memory -- management issues, without losing sight of what the data was being managed for. Most of these successful behaviours can be rephrased as 'making good use of some Smithtown feature' (e.g. responding more often to the prediction prompt).

4.3.2. Other exploration-based ITS's

GUIDON

GUIDON (e.g. Clancey, 1979; 1983) has been very influential program in the field of intelligent tutoring. It guides the interaction of a learner with the well-known MYCIN expert system on infectious diseases, aiming at the communication of underlying domain rules. Although the project is currently developing new ITS-modules on top of a completely restructured expert system (NEOMYCIN), we wi

[22] In this, the Smithtown development team follows Klahr and Dunbar (1988), who argued that scientific investigation involves dual task space problem-solving. Learners may alternate between data-driven problem solving (a search in the space of possible experiments and their outcomes) and hypothesis-driven problem-solving (a search in the space of possible hypotheses and their predictions).

mainly focus on the original MYCIN-GUIDON setup. Our interest stems from the fact that GUIDON provides an exploratory environment, in which learners must try to locate a 'faulty component' by asking for data, formulating hypotheses, etc. In this respect, and even though GUIDON does not draw upon a simulation model (of the relevant human physiology), its environment is fairly similar to SOPHIE-II's environment for electronics troubleshooting.

From the learner's perspective, GUIDON enhances the interface to MYCIN with mixed-initiative dialogue facilities. A learner can request MYCIN case data or ask for MYCIN's evaluation of the problem, determine the dialogue context, change the topic under discussion, request assistance (e.g. hints), or convey his or her current knowledge or hypotheses to the system. At appropriate moments, however, GUIDON takes control of the dialogue by, for instance, correcting an error, or by probing the learner for his or her understanding and use of data.

The tutorial principles that make up this mixed-initiative, Socratic dialogue style have been summarized as follows:

"1. *Be perspicuous:* Have an economical presentation strategy, provide lucid transitions, and adhere to conventional discourse patterns.
2. *Provide orientation to new tasks by top-down refinement:* Provide the student with an organized framework of considerations he should be making, without giving away the solution to the problem (important factors, subgoals, size of the task), thus challenging the student to examine his understanding constructively.
3. *Strictly guide the dialogue:* Say when topics are finished and inferences are completed, as opposed to letting the student discover transitions for himself.
4. *Account for incorrect behaviour in terms of missing expertise* (as opposed to assuming alternative methods and strategies): Explain clearly what is improper from the tutor's point of view (e.g., improper requests for case data). This is, of course, more a statement of how GUIDON models the student than a principle of good teaching.
5. *Probe the student's understanding when you are not sure what he knows*, when you are responding to partial student solutions: Otherwise, directly confirm or correct the solution.
6. *Provide assistance by methodically introducing small steps* that will contribute to the problem's solution:
   a. Assistance should at first be general, to remind the student of solution methods and strategies he already knows;
   b. Assistance should encourage the student to advance the solution by using case data he has already been given.
7. *Examine the student's understanding and introduce new information* whenever there is an opportunity to do so." (Clancey, 1983, p. 13).

On the assumption that MYCIN's diagnostic rule base appropriately represents the knowledge to be learned, GUIDON produces very reasonable and coherent dialogue sequences (Wenger, 1987). However, and in spite of the fact that the MYCIN rules have been annotated with canned explanations, learners find these rules difficult to understand, to remember, and to use in their diagnostic problem solving attempts (Clancey, 1983).

This observation has led to the complete restructuring of MYCIN, best described as a *decompilation* of domain knowledge into a form that is more communicable to learners[23] (Wenger, 1987). This evolution echoes one in the SOPHIE project: although SOPHIE-II's quantitative circuit model was an adequate simulation of the real system, as MYCIN's domain knowledge base was adequate for expert-level diagnostic performance, both of these 'subject-matter representations' turned out to be inappropriate for instructional purposes. The fundamental lesson to be learned is that, if the domain knowledge representation is not designed with *cognitive fidelity* in mind, even the most sophisti-

[23]To clarify the essence of such *epistemological mapping*, Wenger restates Clancey's point that "...decompilation does not necessarily reflect the way knowledge is actually used by experts in their daily practice. Rather, it constitutes a viewpoint on their understanding to which they can resort in dealing with novel situations. This level is useful for communication because it deals with knowledge in such a way that it can be integrated in a general model of the domain." (Wenger, 1987, 277).

cated instructional strategies will be severely limited.

ACT*-based tutors

The tutoring systems built by Anderson and colleagues (e.g. Anderson et al., 1987) incorporate several instructional principles based on cognitive theory, particularly the ACT* theory of human cognition (Anderson, 1983). These tutors deal with the domains of Lisp programming, and of generating proofs in high school geometry or algebra. The main reason they are of interest here is that these domains involve problem-based exploratory learning.

As with QUEST, the instructional features of ACT*-based tutors are very much tied in with cognitive theory, and have therefore already been discussed in Section 4.1. Principles derived from ACT* have also been mentioned in the context of IMTS. Instead of restating them here, we would merely like to elaborate upon the importance of supporting the learner's goal orientation.

First of all, ACT* suggests that a tutor should *make the existing goal hierarchy explicit to the learner*. Having a 'blueprint' of the target performance will help the learner to (a) interpret expert demonstrations, (b) guide his/her own performance attempts as a plan, and (c) interpret expert feedback on these attempts (Gott, 1989). This is mainly relevant if the learning goals involve normative sequences of operations (i.e. operational knowledge), as is the case in the domain of programming or proof generation. The Lisp Tutor, for instance, communicates the skill of writing CDR-recursive functions in terms of a hierarchical plan, even though its structure does not fully correspond to the resulting code syntax. A dedicated 'goals window' is used to display what has been done so far, and what goal is currently being worked on.

Secondly, a tutor should *help the learner to proceed with problem solving*, without giving away complete solutions. In a study of human Lisp tutoring, McKendree et al. (1984) observed that tutors frequently provide hints or suggestions concerning the next goal in the problem solving process. They do so by means of 3 different strategies:

1. *Goal decomposition*, either by explicitly suggesting a goal or by asking leading questions, can prevent a learner from floundering at the initial stage of problem solving.
2. *Reminding* the learner of previous solutions can help to construct a similar plan (i.e. goal hierarchy) for the current problem.
3. *Simplification* can help learners who get stuck at some point in, or are intimidated by a complex problem.

Goal setting behaviour can thus be supported by a tutor in various ways. In the Lisp Tutor, learners are frequently presented with a menu of possible steps to take next, including an option of having the tutor choose. Apart from helping learners to proceed in the first place, such goal support is of major importance to what is being learned. This is so because, according to current theories of learning (e.g. Anderson, 1987; Newell & Rosenbloom, 1981), a newly learned rule will only be evoked in goal contexts similar to the one in which it arose. Learning the right things in the wrong goal context can lead to inert knowledge, that is, to knowledge that will not be retrieved at appropriate times (Bransford et al., 1989). The main point is that learning in the problem-solving context, an often claimed benefit of simulation-based instruction, will not be achieved by just putting the learner in a simulated task environment. Learning in the problem-solving context is above all *learning by applying knowledge in the appropriate goal context*, so that newly learned rules are more likely to be used in similar contexts later on. A tutor should help the learner in getting a picture of the existing goal hierarchy and, if necessary, in setting appropriate goals for further problem solving.

EUROHELP/FysioDisc Coach

The recently finished Eurohelp ESPRIT project, aimed at the design of intelligent help facilities for computer applications, has among other things resulted in a generic coaching system (e.g. Breuker et al., 1987; Breuker, 1988; 1990). While this system has originally been applied to domains such as Unix Mail and the Unix Vi editor, its generic nature has allowed it to be ported to the rather different domain of physiotherapy (Winkels et al.,

1989). The FysioDisc system trains learners in physiotherapeutic diagnosis, based on videodisc records of patient behaviour.

The coaching system used for these different domains is of interest here since they all require a high degree of learner initiative. In the case of computer applications, learners work within the actual application environment (such as Mail), in which they have to solve all kinds of 'on the job' problems in order to proceed. In the case of the FysioDisc system, learners can request investigations (recorded on videodisc), formulate hypotheses, ask for further information, and finally come to a conclusion.

In the EUROHELP context, coaching should not get in the way of ongoing task performance. It is therefore preceded by an analysis in which the help system tries to determine what the user is actually trying to do, and whether he/she is doing it efficiently, inefficiently, or plain wrong. Inefficiency is determined relative to what the learner should be able to do according to the learner model. This model is also used to determine whether there is an occasion for expanding the learner's knowledge, instead of remaining silent or giving operational help only. If a user does something wrong, the learner model may for instance indicate that this particular learner is not yet ready for a deep explanation of why it was wrong, and may benefit more from just being told what the right procedure is.

In the FysioDisc system, which is solely used for instructional purposes, the learner's diagnostic performance is primarily matched against an ideal diagnostic sequence. As with IMTS, FysioDisc asks the learner to specify his or her set of current hypotheses. The Coach can decide to remediate a lack of knowledge or misconception, to remind the learner of something encountered before, to give additional feedback on the state of the problem or the system's actions, or to give positive feedback when the learner performs correctly.

SHERLOCK

SHERLOCK is a computer-coached practice environment for troubleshooting a piece of avionics troubleshooting equipment (Lesgold, Lajoie, Bunzo & Eggan, 1988; Lajoie & Lesgold, 1989). US Air Force technicians use a complex device called a *test station* to test for and locate faults in F-15 electronic navigation equipment. Every now and then the test station itself fails, and technicians find it hard to deal with such failures. As the job routine does not allow for sufficient on-the-job practice on test station troubleshooting, SHERLOCK has been designed to afford pedagogically sound practice on these non-routine failures.

SHERLOCK's learning goals address the basic mental model of an electronic test setup (generic conceptual knowledge), the associated troubleshooting strategies (generic operational knowledge), as well as more domain-specific knowledge about the test-station (conceptual knowledge) and the task structures for troubleshooting this piece of equipment (operational knowledge). Although SHERLOCK simulates the task environment, such as available schematics and measurement devices, it has no runnable device simulation to drive its operation[24]. Nevertheless, from an instructional point of view it is highly relevant to the kind of simulation-based learning we are interested in (de Jong, this volume).

SHERLOCK has been designed from a *cognitive apprenticeship* perspective on education and training (Lajoie & Lesgold, 1989). Cognitive apprenticeship has been advocated as a successful model for the training of real-world tasks, as well as for education in the conceptual knowledge underlying such tasks (Collins, 1988; Collins, Brown & Newman, 1989; Gott, 1989). We will return to it in a later section, and for the moment it is sufficient to mention the major characteristics of this model: learning knowledge and skill in realistic settings (situated learning), supporting the learner so that s/he can still engage in realistically complex tasks (scaffolding), gradual withdrawal of such support when the learner is up to it (fading), and careful sequencing of learning activities.

For learner support, the SHERLOCK development team decided to "...focus on having the envi-

[24]Instead, it has knowledge about "reasonable" or otherwise "expected" goals, plans, and actions for each problem case in the curriculum. The system uses these predesigned troubleshooting trees to generate all system responses (measurement values, hints on how to proceed, etc.).

ronment reveal its structure rather than on having the coach know exactly what to say to a particular student" (Lesgold et al., 1988, p. 4). An example of 'environmental structure' is that learners must usually specify what they want to do through menu hierarchies that are designed to help the learner in structuring the problem-solving process in a fruitful way. Within the constraints of these menus, learners are allowed to pursue any problem-solving path that domain experts deemed relevant in advance, including paths that do not lead to a solution.

Learners need to work through a fixed sequence of problem cases. Instead of individualization at the curriculum level, SHERLOCK adapts to the learner through its main support feature: the hint. At each predefined move in the problem space of a particular troubleshooting exercise the learner can be supported by means of a series of increasingly specific -- and thus increasingly helpful -- hints. A rather global hint would for example be:

> The Test Package provides a path for the Test Station Stimulus signals to get to the Unit and provides a path for the UUT signals to get to the Test Station.

A more specific hint would for example add the following to the above text:

> ... Is there anything different between the TO Test Step and the previous Test Step?

The concept of a hint is a fairly broad one here, since SHERLOCK can give hints of various types. An Action hint suggests a next test and how to carry it out ("...you can now measure the input at P6 Pins S and T with the O'scope"), an Outcome hint helps to get the results of a test ("The signal at P6 Pins S and T is +5VDC, not 25 Hz"), a Conclusion hint helps to determine the meaning of a test result ("In the TO Test Step you set up the Manual Signal Generator so that it would output 25 Hz"), and an Option hint helps to decide what to do next at a higher level than a specific test ("Maybe you should swap the drawer"). These examples, all taken from Gott (1989), show that hints can not only look forward to what can be done next, but can also look back on what has been done so far (e.g. by means of a recapitulation, or an explanation of test significance).

SHERLOCK can handle up to five different levels of hint specificity (most of them 'canned'), and it chooses amongst them on the basis of expected learner performance in the current part of the problem space. Apart from a few exceptional occasions (e.g. truly redundant actions), SHERLOCK only gives a hint when the learner asks for it. This is part of the development team's philosophy of assigning responsibility to and inducing an active attitude in the learner. Learners can ask for a hint in two different ways: pressing 'Help' or 'Panic button!!' in a fixed control menu. While pressing 'Help' results in any of the four types of hints described above, pressing 'Panic button!!' yields a top-down overview of the problem space, showing the areas that have been ruled out as well as those that are still to be explored.

A more general instructional characteristic of SHERLOCK, and one that is intimately related to the cognitive apprenticeship model, is that the system provides holistic instruction. Learners are always engaged in a *complete* real-world task rather than some subtask decoupled from its context, because they are supported ('scaffolded') in carrying out parts of the task they cannot yet handle on their own. Note that this seems to be at variance with common recommendations in 'classic' Instructional Design theory (e.g. practice for mastery of prerequisite subskills before getting to the target skill).

Lajoie and Lesgold (1989) describe a few features that are planned for the next version of SHERLOCK. A very interesting one is to give learners an explicit overview of their solution trace in the problem space at the end of a problem, and to allow comparison with an expert trace by means of an overlay technique. Such articulation or 'reification' of normally unobservable entities is consistent with the cognitive apprenticeship model (e.g. Collins & Brown, 1988), and serves to induce reflection on one's own performance. Another feature scheduled for development is to use common interface techniques (e.g. graying out items) in order to highlight particularly relevant areas of, or paths in, circuit schematics.

Several formal evaluations of SHERLOCK have shown to be successful. One study showed that the approximately 20-25 hours of SHERLOCK training yield a similar troubleshooting proficiency as four

years of on-the-job experience[25]. Other, more qualitative analyses have also shown that practice on SHERLOCK leads to relevant troubleshooting expertise much more rapidly than on-the-job performance, and that this effectiveness holds for trainees of various aptitudes.

Lajoie and Lesgold (1989) identify the following instructional principles as candidate contributors to SHERLOCK's success:

1. Teach specific domain knowledge along with metacognitive skills and heuristics that increase domain knowledge.
2. Embed learning within a problem-solving environment that reflects the uses to which knowledge will be put.
3. Provide opportunities for learning through a combination of guided practice and locally situated opportunities for observations (e.g. expert performance).
4. Emphasize holistic procedural training, and support the learner through the parts he or she cannot yet handle.
5. Train the specific competence required rather than training abstractions; the latter must be grounded in prior concrete experience.
6. Base support on solid cognitive task analysis that identifies critical concepts and procedures, and use the vocabulary of the workplace.
7. Build from the highest levels of domain expertise, that is, on truly experienced people rather than your own model or some job doctrine.
8. Enable the development of complex subgoal structures.
9. Adapt instruction, e.g. by giving coaching only when it is needed.

Although it is not clear what the relative contribution (if any) of each of the above principles is, the whole package has been derived from the cognitive apprenticeship model. Thus, the effectiveness of SHERLOCK lends some overall support to the cognitive apprenticeship paradigm.

4.3.3. Effectiveness of existing systems

As with most ITS's, there is relatively little hard evidence on the effectiveness of the instructional features described above. Much of the evidence up until now is anecdotal, some systems are still under evaluation, and for only a few systems (Smithtown, Lisp tutor, SHERLOCK) have experimental results been reported.

SOPHIE-II was actually used for a short course (Wenger, 1987), but experiments revealed the fundamental restrictions imposed on instruction by a quantitative simulation model. Evidence on the effectiveness of STEAMER has only come from 'anecdotal reports of favourable reactions' from Navy trainees (Gott, 1989). An informal experiment with QUEST showed improved performance on a paper-and-pencil test, and more advanced troubleshooting behaviour when working with the system itself (White & Frederiksen, 1990), while Gott (1989) mentions the 'successful application' of causal model progression in several other domains. Informal reactions on the use of MACH-III in radar mechanics training have been favourable so far, although both learners and instructors appear to pay little attention to its tutoring functionality (Critiquer and Advisor), thereby reducing the system to a fairly plain radar simulation (Kurland, 1989). The IMTS helicopter application is still in experimental evaluation at a military maintenance training school (Towne & Munro, 1989). Experimental evaluation of Smithtown revealed that, compared to 11 hours of traditional classroom instruction, learners needed about 5 hours of simulation work to obtain a similar level of micro-economics understanding (Shute & Glaser, 1990).

Like SOPHIE-II, GUIDON turned out to be limited because of the low cognitive fidelity of its underlying knowledge base: learners had problems with understanding, remembering, and actually using the MYCIN rules (Clancey, 1983). Experiments with the Lisp Tutor have shown that, while

[25] This result should of course be interpreted relative to the time that technicians actually spend on the target task of troubleshooting the *test station*. As mentioned before, test station failures occur infrequently, and the technicians' regular job is to use it in troubleshooting F-15 equipment. Thus, only a minor part of that four year period is time on task as far as this comparison is concerned. Nevertheless, even a low estimate of only 1% time on task (and an assumed 2000 hours working year) would add up to about 80 hours of on-the-job target practice in four years, which is still 3-4 times as much as SHERLOCK practice time.

the human tutor is generally superior to other pedagogical alternatives, the computer tutor is not far behind (Gott, 1989; Shute, 1990). Both the EUROHELP and FysioDisc systems have not yet been assessed on their instructional effectiveness. Experimental evaluation of SHERLOCK, finally, has revealed to be much more efficient than on-the-job training (Lajoie & Lesgold, 1989).

None of these systems has been evaluated in a way that allows for the differential assessment of the effectiveness of various instructional features. However, many of the features described have been derived from cognitive theories (e.g. ACT*), which are in turn corroborated by fundamental cognitive research. Nevertheless, the need remains to test their application in the domain of simulation-based ITS.

Most of the above features are also in line with the well-established principles of traditional apprenticeship instruction. The present state of computer-based instructional technology allows us to return to this resource-intensive mode of instruction, but now for complex real-world tasks which draw more on cognitive than on physical skills (Collins, 1988; Collins, Brown & Newman, 1989; Gott, 1989). Collins discusses six characteristics of *cognitive apprenticeship instruction*:

1. *Situated learning*: "...learning knowledge and skills in contexts that reflect the way the knowledge will be useful in real life" (Collins, 1988, p. 2).
2. *Modelling and explaining*: "...showing how a process unfolds [and] giving reasons why it happens that way" (p. 4), which can either involve processes in the world or the performance of an expert.
3. *Coaching*: providing active guidance and help when needed, and gradually diminishing the degree of assistance as learning progresses (fading).
4. *Reflection on performance*: e.g. by means of replay facilities.
5. *Articulation*: "...forcing students to explain and think about what they are doing" (p. 10), e.g. through the construction of and experimentation with artifacts based on their theories or ideas.
6. *Exploration*: "...pushing students to try out different hypotheses, methods and strategies to see their effects" (p. 12).

In similar vein, Gott (1989) mentions *situated learning*, *scaffolding (external support)*, and *fading* as hallmarks of apprenticeship, and adds a *careful sequencing* of learning activities to the list.

The apprenticeship framework seems particularly relevant to the design of intelligent simulation learning environments (see van Berkum, 1991). Also, one of the few systems that turned out to be clearly successful, SHERLOCK, has been designed with cognitive apprenticeship in mind (Lajoie & Lesgold, 1989). Nevertheless, and although the apprenticeship tradition goes back a long way, its effectiveness in simulation-based ITS geared towards complex cognition remains a subject for research.

5. Instructional environments: a theme-centred approach

In the present paper we have described many kinds of 'instructional measures' taken from a variety of sources: ITS's, use of simulations, instructional design theory and cognitive theory. The present section will summarize and reorganise a selection of the results from the preceding discussion by classifying instructional measures according to the four themes we identified as being important for instructional use of simulations (see de Jong, this volume). In doing this we will add references from literature that did not fit in directly with the survey we made in the preceding sections.

In creating this classification it appeared that specific instructional measures could have multiple functions and thus be classified under different themes. If so, we have placed them only under the heading to which they were most relevant.

5.1. *Theme 1: Domain models*

As a first characteristic of simulations we stated that a simulation contains a *computational model*. Most of the time these models are rather *complex* and have a complicated *quantitative character*. As we have seen in van Joolingen and de Jong (this volume) models can be characterized by specific *epistemological entities and relations* (see also de

Jong, Tait & van Joolingen, 1990; van Joolingen & de Jong, 1991a).

These characteristics of models have led to instructional recommendations of the following kinds:

- providing multiple views of the same model;
- progressive implementation of models;
- offering domain information in a more direct instructional way, additional to the simulation.

a. *Multiple views*

The domain information present in the simulation model can be offered to the learner in a number of ways. A number of studies see the offering of multiple views of the same model as a way to enhance learning. A number of variations exist:

- The issue of *multiple viewpoints* or *perspectives* on the same model is quite central in White and Frederiksen's (1987a,b; 1989; 1990) work on QUEST (see Section 4.3.1). Here different perspectives (a functional, behavioural and physical perspective) of an electrical circuit are offered to learners (see also Section 4.1.1). Offering multiple viewpoints (perspectives) on the same model is also present in STEAMER and MACH-III (see Section 4.3.1).

  Kamsteeg, de Jong and de Hoog (1990) offer a detailed overview of the 'viewpoint' idea.
- The fact that models are mostly complex and numeric may lead to introducing *qualitative descriptions* of basically numeric relations. Moreover, it is assumed that genuine (expert) understanding of a domain is accompanied by having qualitative models of the domain. Plötzner, Spada, Stumpf and Opwis (1990), for example, have designed a microworld for teaching about the physics topic of collision in which they distinguish three domain levels: a semi-quantitative-relational level, a quantitative-relational level and a quantitative-numerical level. In their view learners will traverse these levels from the semi-quantitative-relational level, through the quantitative-relational level to a quantitative-numerical level. If feedback is given to the learner it should be at the appropriate level. This principle can also be found in STEAMER amongst others, where different abstraction levels of the same device are available to the learner (see Section 4.3.1).
- Giving the learner a structured insight into the model might be reached by offering information on the *epistemological* qualities of the model. We are not aware of studies where learners are offered information on epistemological qualities of the model. One ongoing study (Njoo & de Jong, 1991c) offers students who use a computer simulation in mechanical engineering information on (amongst other things) the role of variables and parameters.
- A related principle is described by Reigeluth and Schwartz (1989, Section 4.2.6). They recommend starting an instructional session that includes simulation by presenting a *demonstration mode*, in which no learner activity is involved, before continuing with a *discovery mode*. Demonstration modes are also present in IMTS and MACH-III (Section 4.3.1).
- A second principle related to multiple views is showing relations in the model *in different (diagrammatic) ways* (diagrams, bargraphs, functions etc.). For an example see MACH-III (Section 4.3.1).

b. *Progressive model implementation*

One of the ways to offer complex models is to offer a specific sequence of models (or sub models). The difference between this and offering multiple views, is that in multiple views in principle the *same model* is offered, whereas in progressive model implementation, a sequence of *different models* is presented. The principle is to start with the model that is closest to the initial knowledge of the learner, or to present anchoring points. From this (easy, simple etc.) starting point, a gradual introduction of models that more closely approach the target model is pursued. Here also, we find a number of variations:

- Progressive model implementation, in the sense of an *increasing complexity* of models by adding elements and relations is nicely represented in QUEST (White & Frederiksen, 1987a,b, 1989, 1990). QUEST not only offers learners multiple perspectives of the same model, but also presents these in a sequence of increasing complexity (Section 4.3.1).

Elaboration theory (Reigeluth and Stein, 1983) also prescribes a sequencing from simple to complex, by starting with the shortest procedure and gradually introducing alternative paths (Section 4.2.5).

Simplification is also a feature of ACT*-based tutors, but there it is only used when a learner gets stuck and it is not a part of some sequencing strategy (Section 4.3.2).

- In IMTS (Towne et al., 1988, see Section 4.3.1) the learner is offered exercises that have an increasing level of *difficulty*. This level of difficulty is predetermined, but the actual sequence of offering exercises is set dynamically in the instructional session. An increasing level of difficulty can also be found in MACH-III (Section 4.3.1).
- A specific type of model progression has been proposed in Reigeluth's Elaboration theory (Reigeluth & Stein, 1983; Reigeluth & Schwartz, 1989). They state that when conceptual knowledge is an instructional goal, it is best to start by presenting some *central, general principles* and then proceed by introducing the less central concepts and principles (see Section 4.2.5).
- Not really a sequencing of information principle, but related to it, is Gagné and Brigg's (1979) recommendation to start with the recall of *prerequisite knowledge* (Section 4.2.2).
- The possibility of working with a *submodel* in a complex model is not a sequencing principle as well, but it also offers an opportunity of mastering more complex models by decomposing them into simpler parts. This feature can be found in SOPHIE and STEAMER for example (Section 4.3.1).

c. *Offering additional information*

Several studies point out that effective learning with a simulation decisively depends on the learner's prior knowledge of concepts and sustaining information that is present or related to the model in the simulation. By surrounding or alternating the simulation with more tutorial like instruction, potential problems with insufficient prior knowledge can be overcome.

Instructional design theories mostly take this aspect into account (see Section 4.2) and recommend starting a simulation session with a tutorial-like introduction. Reigeluth and Schwartz (1989) for example recommend starting the simulation session with what they call a 'prototypical example' that learners can observe, in order to prevent them from trial-and-error behaviour when they are in the exploratory phase. An example of offering a tutorial before the simulation can be found in de Jong, de Hoog and de Vries (1991).

A second possibility is to remediate the learner's knowledge while s/he is working with the simulation. This can be done at the learner's initiative (s/he is offered a tutorial or data base (possibly hypermedia-based)) or it can be under the control of the system. An example of this can be found in Shute (1990a) who used an ITS on electricity. Learners had to solve problems with a simulation environment and received domain feedback when making mistakes. The feedback could be of two kinds: deductive and inductive. In the deductive condition learners were given a principle from the theory that applied to their mistake. In the inductive condition, learners were only provided with the relevant variables in the problem.

A way of offering additional information that is nicely integrated with the use of simulations is *providing explanations* of the behaviour of the model. QUEST (see Section 4.1.1.) and MACH-III (Section 4.3.1) are examples of systems offering this type of instructional support.

Finally, Elaboration theory (Section 4.2.5) recommends presenting the learner with additional information *after* an instructional session, by means of summarizers and synthesizers.

5.2. *Theme 2: Learning goals*

At the start of this paper we distinguished three types of learning goals. The first two (*conceptual and operational knowledge*) were referred to as kinds of knowledge, and as a third kind we distinguished *knowledge acquisition knowledge*. Instructional actions from a cognitive theory point of view were extensively discussed in Section 4.1 and an extension with regard to conceptual knowledge has already been given in Section 5.1. Therefore, in the present section, we will only provide a short overview and give some additional information.

a. *Conceptual knowledge*

Quite a few studies stress that computer simulations are not very efficient for teaching *simple* conceptual knowledge. They even sometimes claim that only conceptual knowledge where time is involved (*dynamic models* in which the model contains a process) are suitable for being taught by means of simulations (see for example Reigeluth & Schwartz, 1989).

If the models to be taught are complex, the principles that we discussed above, such as *providing multiple views*, *progressive implementation of models* and *offering additional information*, come into play.

b. *Operational knowledge*

Anderson's theory (e.g. Anderson 1983), which is developed in the context of operational knowledge (such as geometry proofs) recommends that learning takes place in a problem solving context, that model tracing takes place and that there is a minimisation of working memory load (see Section 4.1.2).

The theory also recommends giving immediate feedback when a learner makes a mistake. Munro, Fehling and Towne (1985), however, point out that this is not an instructional strategy to use when the learning goal involves what they call *dynamic skills*. In the latter case, where the learner's attention is heavily consumed by the task itself, it seems a better idea to use '*less-intrusive* feedback', meaning that the learner is presented with a signal that an instructional message is available but is able to neglect or ask for this message.

Tennyson and Elmore (1990) use a computer simulation to teach what they call 'contextual knowledge'. This is *when* and *how* to use operational knowledge and conceptual knowledge. The conceptual knowledge itself is taught in a more expository part of their program.

An instructional approach that is very relevant for teaching operational knowledge is the *cognitive apprenticeship approach*. A discussion of this approach has been given in Section 4.3.3.

Finally, we should stress that learning conceptual knowledge may be very much related to learning operational knowledge. For learning operational knowledge it seems wise to have a thorough understanding of the associated underlying (conceptual) knowledge (see e.g. the discussion on STEAMER in Section 4.3.1 and also Section 4.1.2). Sometimes, therefore, it is recommended that conceptual knowledge be taught first, and possibly by other means than a simulation (see also Sections 4.1.2 and 5.1).

c. *Knowledge acquisition knowledge*

Knowledge acquisition knowledge refers to knowledge the learner has about how to act in order to gain knowledge. In the context of simulations, this refers to proficiency in exploratory (or scientific discover) learning.

In the present section we discuss the instructional measures that can be taken when exploratory learning skills are explicit learning goals of instruction. In Section 5.3 we will discuss instructional measures to support exploratory learning in the case these learning processes are the *vehicles* through which other goals (conceptual or operational knowledge) are to be reached. Inevitably, some overlap will arise.

Within the training programs for teaching 'scientific skills' (see for example Germann, 1989 and Leonard, 1988), we do not find that many programs are developed in the context of learning with a computer simulation.

An exception to this can be found in Friedler, Nachmias and Linn (1990). Their microcomputer based laboratory is explicitly designed for teaching scientific reasoning skills. According to them scientific reasoning is composed of:

- defining a scientific problem;
- stating a hypothesis;
- designing an experiment;
- observing, collecting, analyzing, and interpreting data;

- applying the results;
- making predictions (Friedler et al., 1990, p. 173).

For training these different abilities Friedler et al. use so-called context-free computer games, outside the context of the laboratory. Students are then offered computer games that help them develop (for example) their ability to control variables. Friedler et al. apply this 'context-free' technique, although they themselves repeat the general notion that general abilities are only well learned in the context of a specific domain.

Another study in which scientific discovery skills are an explicit instructional goal, and a computer simulation is used, is presented in Shute and Glaser (1990, see Section 4.3.1). The system discussed (Smithtown) follows the learners' exploratory behaviour and provides feedback on this exploratory behaviour when it detects deficiencies from a predefined ideal exploration pattern (for example when a learner tries to change more than one variable at a time).

Njoo and de Jong (1991c) supported the scientific reasoning process of students' learning with a computer simulation in control theory, by offering them forms that consisted of a number of blocks, each dedicated to a specific phase of the exploratory process. They distinguished model exploration, stating hypothesis, designing an experiment, making a prediction, data interpretation and reinterpretation of the model. Students worked through a simulation using these forms and were urged to fill in every block of the form. Before the simulation started, students were given information about the different phases of the exploratory process, either in general or in domain specific (control theory) terms. In this way both the learning of scientific discovery skills and the attainment of domain specific learning goals is supported.

Glaser and Bassok's (1989) work revealed that learning knowledge acquisition skills is better accomplished when learners work in groups and are able to comment on each other's ideas. This approach comes quite close to the practical situation of learning with computer simulations in which learners quite often work in pairs (see for example Njoo and de Jong, 1991a,b,c).

Also, in Tennyson and Breuer (1990) computer simulations are explicitly used to train 'higher-order thinking skills'. This implies that the domain involved in the simulation is not really important but instrumental to learning the knowledge acquisition skill. Tennyson and Breuer therefore recommend assuring that the learners are very proficient in the domain chosen. For the simulation itself they state, among other things that it should be complex enough, expose students to alternative solutions, allow the students to see consequences of their choices, and that the simulations contained a situation that would collapse without decisions by the learners (e.g. firms go bankrupt). Finally, Tennyson and Breuer recommend the use of group discussion when working with the simulation.

5.3. *Theme 3: Learning processes*

Exploratory learning processes are not the learning goals themselves but form the means of attaining other learning goals. Exploratory learning processes can be sustained by offering instructional *guidance* (directive support) or can be supported by *non-directive support* (e.g. hypotheses scratchpads). Although the difference is sometimes vague, directive support is discussed in the present section, and non-directive support is described in de Hoog et al. (this volume, see also de Jong, 1990).

There seems to be a contradiction between the terms 'exploratory learning' and 'instructional strategy'. Through instruction, initiative is taken away from the learner and put into the hands of the tutor. In the context of exploratory learning we should therefore regard an instructional strategy as a means of guiding exploration, which implies guidance of learning processes characteristic of exploratory learning (see Goodyear et al., this volume).

Basically, this guidance can be provided in two different ways. The first way is to *provide favourable conditions*, so that potential obstacles for learning processes are taken away (and this kind of support is very close to *non-directive support*, of

course). Secondly, learning processes can be *stimulated* explicitly.

a. *Providing favourable conditions*
The instructional strategy may give prescriptions that set an environment in which learning processes and activity can take place. As an example we may think of a strategy of progressive implementation in which the learner starts with a model that lies within his/her capabilities (e.g. White & Frederiksen, 1990 and Section 5.1). This strategy is related to the principle of mastery learning (see e.g. Glaser & Bassok, 1989, p. 639). A second example can be found in Böcker et al. (1989). In their simulated electronics laboratory (ELAB) the learner is not allowed to fill in values for variables that will lead to (for these learners) unexpected and hard to interpret results. In this way, by setting boundaries to the input, the learner is prevented from engaging in erroneous behaviour. A similar approach can be found in Bender (1989) in which the learner gets error messages and prompts to retry if s/he enters values for parameters outside a permitted range.

Another way to prevent a lack of relevant domain information from hindering the learner in exploratory learning is to give additional information. In IMTS (Towne et al., 1988), for example, the learner is given domain information if s/he appears to be stuck, and is not able to continue with the exploratory process. An example of such a 'problem hint' is: "K107 must be energized to have power to the Spread/Fold circuit" (Towne et al., 1988, p. 27). Similar types of hints, in which domain information is given, can be found in SHERLOCK (see Section 4.3.1). On this topic see also Section 5.1.

b. *Stimulating learning processes*
A second way of supporting learning processes and activity is explicitly stimulating them. Again there are a number of options:

- The use of a *Socratic dialogue* in order to have the learner revise his/her thoughts is a widely spread technique. Brna (1987), for example, created a microworld on the physics topic 'dynamics' in which learners are confronted with the results of their own misconceptions. A second example can be found in Finegold and Gorsky (1989). The basic idea in their study is the creation of 'cognitive dissonance' and helping the learner to overcome this. Classroom discussions are less well suited for this. Computer simulations, according to the authors, are an excellent medium. Their domain is mechanics and the situation presented is a book on a table. The question is: which forces are acting? The learner has to reason himself out of a cognitive dissonance (e.g the learner who has indicated that there are no forces on the book is shown that the book will stay at the same place if the table is taken away).

  Junck and Calley (1985) developed a genetics laboratory in which problems are posed by the computer and solved by the learner and the tutor in cooperation. They state that in this method (which they call the 'post-Socratic pedagogy') the tutor is encouraged to "help students solve problems by either suggesting heuristics or by pointing out certain patterns in the data" (p. 14).

  This teaching process of 'Collaborative learning' is also advocated by Chan and Baskin (1988) and Behrend, Singer and Roschelle (1988). Studies in which learners are not explicitly encouraged to activate their prior knowledge, show that learners might learn how to play a game with the simulation, but fail to understand essential relationships in the model (Flick, 1990).

  Finally, we have seen that Collins and Stevens (1983, see Section 4.2.4) ask learners who have been shown to possess a misconception, to make deductions from that misconception that they will probably not agree with.
- Collins and Stevens (1983, see Section 4.2.4) stress in their work the *systematic variation of cases*. This variation is not meant as a form of progressive implementation, but to encourage the learner in stating model relations. Systematic variation is also meant for *testing hypotheses*.

- Providing *hints and suggestions* on how to perform exploratory learning is yet another way to elicit learning processes. Rivers and Vockell (1987) provided learners with general guidelines on how to work with a simulation. Examples of these guidelines are: 'It is a good idea to test only one variable at a time', and 'Look for patterns or relationships as you systematically change the variables'. They found that learners who received these guidelines showed better knowledge acquisition behaviour than learners who did not receive these guidelines. Similar types of hints are given in Smithtown (Shute & Glaser, 1990; see also Section 4.3.1).

  Friedler et al. (1990) offered learners specific information on how to carry out 'Observation' and 'Prediction'. The domain was heat and temperature and they gave students forms to note down their hypothesis that already unravelled the elements of the hypothesis. For example: "Predict the temperature at the beginning of the experiment, when the water starts to boil etc. What do you base your predictions on" etc.
- A very common instructional measure that accompanies simulations is giving *goals or assignments* with the simulations. This seems to be a very motivating measure in which learners are given (for example) an *optimisation assignment* (such as 'maximise the profit of a factory'). These measures, however, also tend to have the side effect of inhibiting learning processes as well. Learners will not be inclined to test hypotheses they have by introducing manipulations that will distract them from the assignment (van Joolingen & de Jong, 1991b).
- Offering a *fault diagnosis task* is yet another type of specific assignment. Of course fault diagnosis is used in the simulation when troubleshooting is the procedure to be learned (as in MACH-III, see Section 4.3.1), but it can also be used as a technique to evoke exploratory learning processes with learners (see Gott, 1989).
- Collins and Stevens (1982, Section 4.2.4) have offered a number of *prodding techniques* for encouraging the statement of hypotheses.
- When the learner has some trouble with simulation-characteristic learning processes, it might be a good idea to *take over some processes* from the learner in order to enable other processes. Njoo and de Jong (1991c), for example, offered students in mechanical engineering ready made hypotheses to explore. Here, learners are encouraged to perform other steps in the discovery process (such as setting up an experiment) because they are explicitly supported in one of the most difficult learning processes involved. Taking over difficult processes from a learner is of course closely related to the idea of *scaffolding* as it is used in the cognitive apprenticeship approach (see Section 4.3.3).

5.4. *Theme 4: Learner activity*

Learner activity refers to the *manipulations* the learner is allowed to make when working with the simulation. Learner activity is mostly sustained by providing the learner with (non-directive) support through the interface (e.g. by facilitating easy change of input variables, see de Hoog et al., this volume). Some of the measures mentioned above (e.g. the hints by Rivers & Vockell, 1987; and those by Shute and Glaser, 1990; or the hiding of variables done by Böcker et al., 1989) not only direct exploratory learning processes but also affect the learner activities associated with these learning processes.

6. Discussion

This paper has reviewed the literature for its potential relevance to the design of simulation-based learning environments. After an examination of learning goals for simulations, we have surveyed the research areas of human learning and cognition, of instructional design, and of intelligent tutoring systems for exploratory learning. This has resulted in a wide variety of instructional support features and related principles. In addition, we have

reapproached the literature from a theme-oriented perspective on simulation-based learning (de Jong, this volume), adding some material that did not readily emerge from the first pass.

Of course, the present collection of instructional support features and principles is not a complete one. As research and field application proceeds, new instructional support options will emerge. More importantly, much needed evidence will come in on the effectiveness of various options.

In our overview, we have tried to describe the options along with their claimed justification, if any, but we have *not* made a selection of preferable support features for simulations (see van Berkum, 1991, for a selection based on cognitive apprenticeship). We are however convinced that *any* choices made during the design of a simulation-based learning environment must be (directly or indirectly) grounded in an assessment of the learning goals at which that environment is supposed to aim. We hope that the proposed learning goal taxonomy will be of help in that assessment.

References

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

Alexander, P.A., & Judy, J.E. (1988). The interaction of domain-specific and strategic knowledge in academic performance. *Review of Educational Research, 58,* 375-405.

Anderson, J.R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson, J.R. (1985). *Cognitive psychology and its implications (2nd edition).* New York: Freeman.

Anderson, J.R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94(2),* 192-210.

Anderson, J.R., & Reiser, B.J. (1985). The LISP tutor. *Byte, 10,* 456-462.

Anderson, J.R., Boyle, C.F., Farrell, R. & Reiser, B.J. (1987). Cognitive principles in the design of computer tutors. In P. Morris (Ed.), *Modelling cognition.* Chichester: Wiley.

Aronson, D.T.M., & Briggs, L.J. (1983). Contributions of Gagné and Briggs to a prescriptive model of instruction. In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 75-100). Hillsdale: Lawrence Erlbaum.

Behrend, S., Singer, J., & Roschelle, J. (1988). A methodology for the analysis of collaborative learning in a physics microworld. *Proceedings of ITS-88* (pp. 48-53). Montreal, Canada.

Bender, D.A. (1989). Combining a computer simulation with a laboratory class - the best of both worlds? *Computers and Education, 13,* 235-243.

Berkum, J.J.A. van (1991). *Functional requirements for an intelligent simulation learning environment: A cognitive apprenticeship perspective* (DELTA-project SAFE Report No. SIM/24B). Zaandam: Courseware Europe bv.

Bloom, B.S (Ed.). (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain.* New York: McKay.

Böcker, H.D., Herczeg, J., & Herczeg, M. (1989). ELAB - An electronics laboratory. In D. Bierman, J. Breuker & J. Sandberg (Eds.), *Proceedings of the 4th international conference on AI & Education* (pp. 15-25), Amsterdam: IOS.

Bransford, J.D., Vye, N.J., Adams, L.T. & Perfetto, G.A. (1989). Learning skills and the acquisition of knowledge. In A. Lesgold & R. Glaser (Eds.), *Foundations for a psychology of education.* Hillsdale, NJ: Lawrence Erlbaum.

Breuker, J. (Ed.) (1990). *EUROHELP final report.* University of Amsterdam, Department of Social Science Informatics.

Breuker, J., Winkels, R. & Sandberg, J. (1987). A shell for intelligent help systems. In *Proceedings of the tenth IJCAI,* August 23-28, 1987, Milan. Los Altos, CA: Morgan Kaufman.

Breuker, J. (1988). Coaching in help systems. In J. Self (Ed.), *Artificial intelligence and human learning: Intelligent computer-aided instruction.* London: Chapman & Hall.

Brna, P. (1987). Confronting dynamics misconceptions. *Instructional Science, 16,* 351-379.

Broadbent, D.E. & Aston, B. (1978). Human control of a simulated economic system. *Ergonomics, 21(12),* 1035-1043.

Brown, J.S., Burton, R.R. & deKleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III.

In D. Sleeman & J.S. Brown (Eds.), *Intelligent tutoring systems*. London: Academic Press.

Burton, R.R. & Brown, J.S. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies, 11,* 5-24. (Reprinted in D. Sleeman & J.S. Brown (Eds.) (1982). *Intelligent tutoring systems*. London: Academic Press).

Chan, T.W., & Baskin, A.B. (1988). Studying with the prince. The computer as a learning companion. *Proceedings of ITS-88* (pp. 194-200). Montreal, Canada.

Clancey, W.J. (1979). Tutoring rules for guiding a case method dialogue. *International Journal of Man-Machine Studies, 11,* 25-49. (Reprinted in D. Sleeman & J.S. Brown (Eds.) (1982). *Intelligent tutoring systems*. London: Academic Press).

Clancey, W.J. (1983). Guidon. *Journal of Computer-Based Instruction, 10(1&2),* 8-15.

Coleman, T.G. & Randall, J.E. (1985). *HUMAN-PC: A comprehensive physiological model* [Computer program]. Jackson: University of Mississippi Medical Center.

Collins, A. (1988). *Cognitive apprenticeship and instructional technology* (Report No. 6899). Cambridge, MA: BBN Laboratories.

Collins, A. & Brown, J.S. (1988). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems*. New York: Springer-Verlag.

Collins, A., Brown, J.S. & Newman, S.E. (1989). Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. In L.B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum.

Collins, A., & Stevens, A.L. (1982). Goals and strategies of inquiry teachers. In R. Glaser (Ed.), *Advances in instructional psychology*. Hillsdale, NJ: Lawrence Erlbaum.

Collins, A., & Stevens, A.L. (1983). A cognitive theory of inquiry teaching. In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 247-278). Hillsdale: Lawrence Erlbaum.

Ferguson-Hessler, M.G.M. & de Jong, T. (1990). Studying physics text; differences in study processes between good and poor performers. *Cognition and Instruction, 7,* 41-54.

Finegold, M., & Gorsky, P. (1989). Learning about forces: simulating the outcomes of pupil's misconceptions. *Instructional Science, 17,* 251-261.

Fitts, P.M. (1964). Perceptual-motor skill learning. In A.W. Melton (Ed.), Categories of human learning. New York: Academic Press.

Fitts, P.M. & Posner, M.I. (1967). *Human performance*. Belmont, CA: Brooks Cole.

Flick, L.B. (1990). Interaction of intuitive physics with computer simulated physics. *Journal of Research in Science Teaching, 27,* 219-231.

Forbus, K. (1984). *An interactive laboratory for teaching control system concepts* (BBN Report No. 5511). Cambridge, MA: Bolt Beranek and Newman.

Friedler, Y., Nachmias, R., Linn, M.C. (1990). Learning scientific reasoning skills in microcomputer-based laboratories. *Journal of Research in Science Teaching, 27,* 173-191.

Gagné, R.M. (1965). *The conditions of learning (1st ed.)*. New York: Holt, Rinehart & Winston.

Gagné, R.M. (1977). *The conditions of learning (3rd ed.)*. New York: Holt, Rinehart & Winston.

Gagné, R.M., & Briggs, L.J. (1979). *Principles of instructional design* (2nd ed.). New York: Holt, Rinehart & Winston.

Gagné, R.M. (1985). *The conditions of learning (4th ed.)*. New York: Holt, Rinehart & Winston.

Gagné, R.M. & Glaser, R. (1987). Foundations in learning research. In R.M. Gagné (Ed.) *Instructional technology: Foundations*. Hillsdale, NJ: Lawrence Erlbaum.

Gentner, D., & Stevens, A. (1983). *Mental models*. Hillsdale: Lawrence Erlbaum.

Germann, P.J. (1989). The processes of biological investigations test. *Journal of Research in Science Teaching, 26,* 609-625.

Glaser, R. & Bassok, M. (1989). Learning theory and the study of instruction. *Annual Review of Psychology, 40,* 631-666.

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.

Groot, A.D. de (1965). *Thought and choice in chess*. Den Haag: Mouton.

Gropper, G.L. (1983). A behavioral approach to

instructional prescription. In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status*(pp. 101-163). Hillsdale, NJ: Lawrence Erlbaum.

Hannafin, M.J., & Hieber, L.P. (1989). Psychological foundations of instructional design for emerging computer-based instructional technologies: 1. *ETR&D, 37,* 91-101.

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.

Hayes, N.A. & Broadbent, D.E. (1988). Two modes of learning for interactive tasks. *Cognition, 28,* 249-276.

Hijne, H. & Berkum, J.J.A. van (1990, November). *A functional architecture for intelligent simulation learning environments.* Paper presented at the DELTA & Beyond conference, The Hague, The Netherlands.

Hijne, H. & Jong, T. de (1989). *SIMULATE: SIMULation Authoring Tools Environment* (OCTO-Report No. 89/02). Eindhoven University of Technology, Department of Philosophy and Social Sciences.

Hollan, J.D., Hutchins, E.L. & Weitzman, L. (1984). STEAMER: An interactive inspectable simulation-based training system. *AI Magazine, 5(2),* 15-27 (also available in G.P. Kearsley (Ed.) (1987), *Artificial intelligence and instruction: Applications and methods.* Reading, MA: Addison-Wesley).

Jong de, T. (1990). *Learning and instruction with computer simulations: interface aspects.* Paper to present at the NATO AETW 'Cognitive modelling and interactive environments', Eindhoven (The Netherlands) November 5-7.

Jong de, T., & Ferguson-Hessler, M.G.M. (1986). Cognitive structures of good and poor novice problem solvers in physics. *Journal of Educational Psychology, 78,* 279-288.

Jong, T. de, Hoog, R. de, & Vries, F. de (1991). *SUPER-MIDAS. A computer simulation for learning about decision support theory.* DELTA project SAFE. Report SAFE/SIM/WPIII/UvA-rep/-SUPER-MIDAS. Amsterdam: University of Amsterdam.

Jong de, T., & Tait, K., & van Joolingen, W.R. (1990). *Formalisation as a step towards specifying tools to support authoring.* Paper presented at the DELTA and Beyond conference, The Hague (The Netherlands), October 18-19.

Joolingen van, W. & de Jong, T. (1991a). An instruction related domain representation for simulations. In T. de Jong & K. tait (Eds), *Towards formalising the components of an Intelligent Simulation Learning Environment.* DELTA project SAFE (P7061). Eindhoven: Eindhoven University of Technology.

Joolingen van, W. & de Jong, T. (1991b). A prototype scratchpad for hypothesis formation and experimental design In: Hijne, H., & van Berkum, J. (Ed.). *Prototype/mock-up of an Integrated Simulation-based Learning Environment.* DELTA project SAFE (P7061). Courseware Europe BV., SAFE/SIM/CE-rep/wp3plans

Jungck, J.R., & Calley, J.N. (1985). Strategic simulations and post-Socratic pedagogy: Constructing computer software to develop long-term inference through experimental inquiry. *American Biology Teacher, 47(1),* 11-15.

Kamsteeg, P., de Jong, T., de Hoog, R. (1990). *Description and classification of domains for open learning environments.* DELTA project SAFE. Report SAFE/SMART/D2/UvA-rep/description. Amsterdam: University of Amsterdam.

Kieras, D.E., & Bovair, S. (1984). The role of mental models in learning to operate a device. *Cognitive Science, 8,* 255-273.

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

Kratwohl, D.R., Bloom, B.S. & Masia, B.B. (Eds.). (1964). *Taxonomy of educational objectives: The classification of educational goals. Handbook II: Affective domain.* New York: McKay.

Kurland, L.C. (1989). Tutor, tool & training environment: Teaching troubleshooting using AI. In D. Bierman, J. Breuker & J. Sandberg (Eds.), *Proceedings of the 4th international conference on AI & Education,* Amsterdam, The Netherlands.

Kurland, L.C. & Tenney, Y.J. (1988). Issues in developing an intelligent tutor for a real-world domain: Training in radar mechanics. In J. Psotka, L.D. Massey & S.A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned.*

Hillsdale, NJ: Lawrence Erlbaum.

Lajoie, S.P. & Lesgold, A. (1989). Apprenticeship training in the workplace: Computer-coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning, 3,* 7-28.

Landa, L.N. (1983). Descriptive and prescriptive theories of learning and instruction: An analysis of their relationships and interactions. In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 55-75). Hillsdale, NJ: Lawrence Erlbaum.

Leinhardt, G. & Greeno, J.G. (1986). The cognitive skill of teaching. *Journal of Educational Psychology, 78(2),* 75-95.

Leonard, W.H. (1988). An experimental test of an extended discretion laboratory approach for university general biology. *Journal of Research in Science Teaching, 26,* 79-91.

Lesgold, A., Lajoie, S., Bunzo, M. & Eggan, G. (1988). *SHERLOCK: A coached practice environment for an electronics troubleshooting job* (Report). Pittsburg: Learning Research and Development Center.

Mager, R.F. (1962). *Preparing objectives for instruction.* Belmont, CA: Fearon.

McKendree, J., Reiser, B.J. & Anderson, J.R. (1984). Tutorial goals and strategies in the instruction of programming skills. *Proceedings of the Sixth Cognitive Science Society Conference,* Boulder, Colorado.

Merrill, M.D. (1983). Component Display Theory. In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status.* Hillsdale, NJ: Lawrence Erlbaum.

Merrill, M.D. (1987). The new Component Design Theory: Instructional design for courseware authoring. *Instructional Science, 16,* 19-34.

Merrill, M.D. (1988). Applying Component Display Theory to the design of courseware. In D.H. Jonassen (Ed.), *Instructional design for microcomputer courseware.* Hillsdale, NJ: Lawrence Erlbaum.

Michalski, R. (1987). Learning strategies and automated knowledge acquisition. In: Bradshaw, G.L., Langley, P., Michalski, R.S., Ohlsson, S., Rendell, L.A., Simon, H.A., & Wolff, J.G. (Eds.) *Computational models of learning.* Bern: Springer-Verlag.

Min, F.B.M. (1987). *Computersimulatie als leermiddel: Een inleiding in methoden en technieken.* Schoonhoven: Academic Service.

Nawrocki, L.H. (1987). Artificial intelligence applications to maintenance training. In G.P. Kearsley (Ed.), *Artificial intelligence and instruction: Applications and methods.* Reading, MA: Addison-Wesley.

Newell, A. & Rosenbloom, P.S. (1981). Mechanisms of skill acquisition and the law of practice. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Lawrence Erlbaum.

Njoo, M., & de Jong, T. (1991a). *Learning processes of students working with a computer simulation in mechanical engineering.* To appear in the proceedings of the 1989 EARLI conference, Madrid.

Njoo, M., & de Jong, T. (1991b). *Stimulating exploratory learning with a computer simulation for control theory: the effect of hints.* Eindhoven University of Technology: OCTO reports (to appear).

Njoo, M., & de Jong, T. (1991c). *The effect of offering study process planning support, learning process information, and ready made hypotheses, on learning with a computer simulation on control theory.* Paper presented at the 1991 AERA conference. Chicago, April 3-7, 1991.

Ohlsson, S. (1986). Some principles of intelligent tutoring. *Instructional Science, 14,* 293-326.

Payne, S.J. (1988). Methods and mental models in theories of cognitive skill. In J. Self (Ed.), *Artificial intelligence and human learning: Intelligent computer-aided instruction.* London: Chapman & Hall.

Percival, F. & Ellington, H. (1988). *A handbook of educational technology.* London: Kogan Page.

Plötzner, R., Spada, H., Stumpf, M., & Opwis, K. (1990). *Learning qualitative reasoning in a microworld for elastic impacts.* Research group on cognitive systems, Forschungsbericht nr. 59. University of Freiburg: Psychological Institute.

Psotka, J., Massey, L.D. & Mutter, S.A. (Eds.). (1988). *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Lawrence Erlbaum.

Reigeluth, C.M. (1983). Instructional design: What is it and why is it? In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 3-37). Hillsdale: Lawrence Erlbaum.

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.

Reigeluth, C.M., & Stein, F.S. (1983). The elaboration theory of instruction. In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 335-381). Hillsdale: Lawrence Erlbaum.

Rivers, R.H., & Vockell, E. (1987). Computer simulations to stimulate scientific problem solving. *Journal of Research in Science Teaching, 24,* 403-415.

Romiszowski, A.J. (1981). *Designing instructional systems: Decision making in course planning and curriculum design.* London: Kogan Page.

Romiszowski, A.J. (1984). *Producing instructional systems: Lesson planning for individualized and group learning activities.* London: Kogan Page.

Rumelhart, D.E. & Norman, D.A. (1981). Analogical processes in learning. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Lawrence Erlbaum.

Schank, R.C. & Farrell, R. (1988). Creativity in education: A standard for computer-based teaching. *Machine-Mediated Learning, 2,* 175-194.

Shute, V.J. (1990). *A comparison of inductive and deductive learning environments: Which is better for whom and why?* Paper presented at the American Educational Research Association (AERA) Annual Meeting, Boston, USA.

Shute, V.J. & Glaser, R. (1990). A large-scale evaluation of an intelligent discovery world: Smithtown. *Interactive Learning Environments, 1,* 51-77.

Shute, V.J., Glaser, R. & Raghavan, K. (1989). Inference and discovery in an exploratory laboratory. In P.L. Ackerman, R.J. Sternberg & R. Glaser (Eds.), *Learning and individual differences: Advances in theory and research.* Ney York: W.H. Freeman & Company.

Stevens, A. & Roberts, B. (1983). Quantitative and qualitative simulation in computer based training. *Journal of Computer-Based Instruction, 10(1&2),* 16-19.

Tennyson, R.D., & Breuer, K. (1990). *Complex-dynamic simulations to improve higher-order thinking strategies.* Paper presented at the AERA, Boston, 1990.

Tennyson, R.D., & Elmore, R.L. (1990). *Acquisition of domain of knowledge: employment of integrated instructional strategies.* Paper presented at the AERA, Boston, 1990.

Towne, D.M. & Munro, A. (1988). The Intelligent Training Maintenance System. In J. Psotka, L.D. Massey & S.A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned.* Hillsdale, NJ: Lawrence Erlbaum.

Towne, D.M. & Munro, A. (1989). Artificial intelligence in training diagnostic skills. In D. Bierman, J. Breuker & J. Sandberg (Eds.), *Proceedings of the 4th international conference on AI & Education,* Amsterdam, The Netherlands.

Towne, D.M., Munro, A., Pizzini, Q.A., Surmon, D.S. & Wogulis, J. (1988). *Intelligent maintenance training technology* (Report No. 110). Redondo Beach, CA: University of Southern California, Behavioral Technology Laboratories.

Towne, D.M., Munro, A., Pizzini, Q., Surmon, D., Coller, L. & Wogulis, J. (1990). Model-building tools for simulation-based training. *Interactive Learning Environments, 1,* 33-50.

Wager, W. & Gagné, R.M. (1988). Designing computer-aided instruction. In D.H. Jonassen (Ed.), *Instructional design for microcomputer courseware.* Hillsdale, NJ: Lawrence Erlbaum.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge.* Los Altos, CA: Morgan Kaufmann.

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. (1989). Causal models as intelligent learning environments for science and engineering education. *Applied Artificial Intelligence, 3(2-3),* 83-106.

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

Williams, R.G. (1977). A behavioral typology of

educational objectives for the cognitive domain. *Educational Technology, 17(6).*

Winkels, R., Achthoven, W. & Gennip, A. van (1989). Methodology and modularity in ITS design. In D. Bierman, J. Breuker & J. Sandberg (Eds.), *Proceedings of the 4th international conference on AI & Education,* Amsterdam, The Netherlands.

Winograd, T. (1975). Frame representations and the declarative/procedural controversy. In D.G. Bobrow & A. Collins (Eds.), *Representation and understanding*. New York: Academic Press.