The Knowledge Engineering Review, Vol. 14:12, 1999, 221–255
Printed in the United Kingdom

Design approaches to model-based simulation in intelligent computer assisted instruction

BJOERN HELFESRIEDER and VENKY SHANKARARAMAN

Department of Computer Science, University of Hertfordshire, Hatfield, UK

Abstract

Model-based simulation systems have been created in various fields of engineering to train personnel or students in operation, maintenance and troubleshooting of complex devices and systems. A review of literature indicates a lack of good overviews of the approaches to system design of model-based training simulations in Intelligent Computer Assisted Instruction (ICAI). Though single systems have to some extent been evaluated with regard to their performance, an organised evaluation, especially a comparative evaluation of the systems that have been created within the field is lacking. To be able to successfully conduct an in-depth review under these conditions, we concentrate and centre our review around the design of the domain simulator component and its implications for the other parts of instructional system. For this purpose we have identified features that are in our opinion suitable to characterise the design approach of a model-based simulation in ICAI for training. We have organised these design features in a framework to enable an effective review. Within this framework, we investigate each feature separately and illustrate it by giving examples from existing applications where appropriate.

1 Introduction

Over the last decade, the face of labour and work has been constantly and rapidly changing. Globalisation in industry and business as well as technological advances have placed different requirements on employees, jobs, and the work process as such. With the development of highly integrated machinery accompanied by fast growing infrastructures such as the Internet the complexity of our world is constantly increasing.

The great variety of different artefacts as well as the speed with which they are replaced by newer, usually more complex ones leads to the strong demand for instruction and training of the personnel operating them. An additional demand arises because of the often limited or unavailable time, money, equipment and human resources. These demands can be effectively met not by one single instructional technology or methodology but by a combined number of technological approaches such as tele-teaching, computer-based training, multimedia technology, and traditional methods such as on-the-job training or classroom instruction. In this scenario, learning is a key element, but not the only one. Training and practising in the sense of applying previously acquired knowledge is needed as well to ensure the comprehensiveness, efficiency and applicability of the acquired knowledge.

1.1 The need for practical training

Different kinds of working environments and tasks lead to different demands on the knowledge the employee needs to perform well on the job. In job settings that require the operation of a complex

system, training is the essential element. Learning about the system can be seen only as the first phase, followed by the second phase of practically applying the previously acquired knowledge.

Complexity in devices or systems can manifest itself in many ways. Complexity can arise through the number of procedures the operator needs to perform, which may be too large for rote memorisation or the need to invent novel procedures when faced with unexpected process states. Additionally, the capability to operate a complex system not only relates to the operation of the system in working order but also to the crucial need to being able to cope with malfunctions and other abnormal system behaviour. It is, therefore, necessary in many tasks not only to simply rote-learn static procedures, but also develop a qualitative understanding of the complex operations of the system. The operator needs to be able to make predictions on possible future device behaviour or infer the causes for faulty behaviour from his or her observations. In most cases, this will include the need for real-time performance which places additional difficulties on the task at hand. Training in such situations should support hypothesis formation and verification, acquisition of expertise on different hierarchical levels, and the formation of mental models about the instructed system or domain (Williams et al., 1983).

1.2 ICAI in training applications

Intelligent Computer Aided Instruction (ICAI) systems are computer-based systems that deal with the problems discussed in the previous section by employing Artificial Intelligence (AI) mechanisms to manage the instructional task, thus providing highly automated and individualised instruction. For example, AI can be employed in connection with the selection of a problem for instruction and the appropriate instructional strategies. It may also be employed to automatically adapt to the student's level of expertise, to trace misconceptions and to provide appropriate remedy.

In literature, ICAI systems are sometimes referred to as Intelligent Tutoring Systems (ITS). It must be noted some researchers argue the difference between ICAI and ITS. For the purpose of this paper, both are considered to be the same. Most of the ICAI systems are similar regarding their general basic building blocks. The commonly accepted components of an ICAI, as put forward for example by Wenger and others (Rickel, 1989; Wenger, 1987), identifies as the main components the student model, instructional module, the pedagogical knowledge responsible for didactic decisions, the domain knowledge component and possibly an interface component (see Figure 1). The domain knowledge component contains expert knowledge about the domain. In most cases it also contains an additional domain model which provides a dynamic training environment that enables the trainee to interact with a simulation of the domain.

Due to the needed resources time and expertise, usually from very different fields, these systems are difficult to construct. Since such a system needs a deep knowledge of a the domain to be instructed for comprehensive and deep instruction, it is especially hard to transfer and, by that,

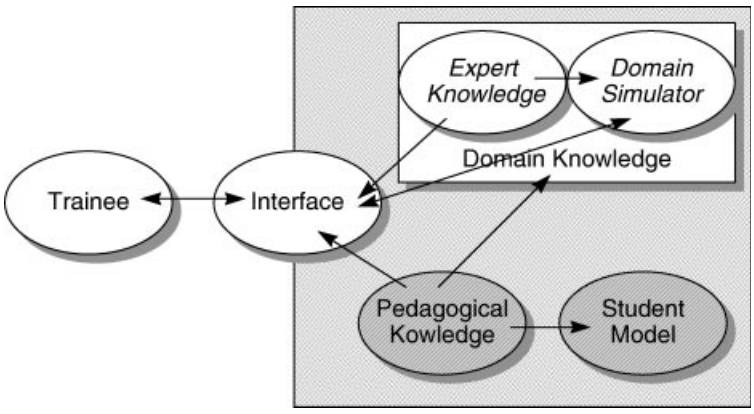

Figure 1 General architecture of a simulation-based ICAI system

reuse parts of an application in one domain to another. Consequently, only a few of these systems have been built to prototype stage, and even fewer have been evaluated or field-tested.

In spite of these problems, results of past and recent ICAI research projects are promising, and show that the approach to tutoring and training is well targeted and justified. Research has shown that automated instruction provided by ICAI systems in some domains can yield better results than instruction using comparable traditional instructional settings (Lesgold, 1992; Lesgold et al., 1994).

1.3 Using models for training

Comprehensive computer-based training on a complex device or system usually involves a model of the device or system as the basis for generating instruction, which is referred to as the "domain simulation". This simulation is the representation of the domain and is an approximate replication of the real device or system. Usually, the simulation is responsive in such that it provides the user with the appropriate feedback for his or her interactions. The simulation model alone is not capable of exercising any kind of concrete instructional functions, such as content selection or tracing student misconceptions during an instructional session. These instructional functions must be accomplished by other instructional components within the system architecture. A detailed description of the role of models in ICAI is presented in section 2.

In ICAI systems developed in the past, models of varying levels of complexity have been used as an integral part to represent the domain or selected parts of the domain (see, for example, Johnson & Rickel, 1996; Vasandani & Govindaraj, 1995; Towne & Munro, 1988; Woolf et al., 1986; Hollan et al., 1984; Brown et al., 1982). These systems have been created mostly in the domain of engineering, probably because training needs are high and models can be formulated somewhat easier than in other domains. Within the engineering domain ICAI systems have been applied to train in operation, maintenance and troubleshooting of complex devices and systems. These include marine power plant operator training, steam plant operator training, troubleshooting of electronic circuits, troubleshooting of military electronic equipment, helicopter bladefolding operation, theorem proofing with qualitative (naïve) physics models, thermodynamic cycle design tutoring and training, tutor and train the operation of a recovery boiler and others.

1.4 Motivation and organisation

A review of literature indicates a lack of good overview of the approaches to systems design of model-based simulations for training in ICAI. Though single systems have to some extent been evaluated with regard to their performance, an organised evaluation, especially a comparative evaluation of the systems that have been created within the field is lacking. To be able to successfully conduct an in-depth review under these conditions, we concentrate and centre our review around the design of the domain simulator component and its implications for the other parts of instructional system.

For this purpose, we have identified features that are in our opinion suitable to characterise a design approach of a model-based simulation in ICAI for training. We have organised these design features in a framework to enable an effective review. Within this framework we investigate each feature separately and illustrate it by giving examples from existing applications where appropriate.

1.5 Overview

Section 2 describes the general motivation for using models of complex systems for training. It provides the theoretical background and basic definitions in the context of simulation and model-based training. The section also gives a short overview of several relevant simulation-based systems in intelligent computer assisted instruction and their areas of application. Section 3 provides the framework of design features which have been identified in order to successfully review the design approaches to model-based simulation in ICAI. The section provides many examples taken from

existing systems to illustrate a specific design feature. Section 4 discusses selected design features which have been described in the previous section. The section also discusses the field in general regarding its achievements and shortcomings. Section 5 summarises the paper, and finally, tries to highlight the major current and future trends and developments as we see them.

2 Using models for simulation-based training

Employing models for computer-based training yields several basic advantages over training on real equipment. These stem for the most part from the ''virtual nature'' of the computer-based model. Additionally, the model offers flexibility and allows a variety of instructional interactions with the trainee.

2.1 Basic issues and needs

Modelling a complex system, and thus having it available as a domain representation for training, requires expert understanding of the domain, computational expertise and is often very time consuming. Although the fidelity of the model and other aspects can be significantly reduced by omitting details of the real equipment, issues such as model correctness and model completeness have to be considered to provide an adequate training scenario.

An operable computer-based model of an equipment yields a number of basic advantages that can justify the effort. The virtual nature of a computer-based training simulation enables training to be accomplished without risking damage to man or in most cases to expensive equipment. Additionally, real equipment as a production resource is often in use and thus not available for training purposes whereas multiple copies of the computer-based system can be produced and installed at minimal cost (Shankararaman & MacFarlane, 1992).

2.2 Characteristics of a model

Any computer model of a real system can be formally characterised by three basic attributes. These attributes describe the relationship of the model to the real system it is modelling (the referent), the dimensions of potential use of the model (the purpose) and cost considerations (the cost effectiveness).

The referent relates to the real world device or system that is to be modelled (Rothenberg, 1989). In this context it should be understood that the model cannot identically copy the referent completely—it can only be a close approximation. The second attribute is the intended cognitive purpose of the model. This purpose can be manifold and for example include comprehension, prediction, manipulation, or simply gaining experience of the referent. The third attribute is the cost effectiveness of the model in relation to its referent. The cost should be considered with respect to: the basic costs of the referent vs. development and maintenance costs of the computer model; the availability of the referent for training vs. possibility of training on multiple instances of the computer model; safety aspects of training on the referent vs. training on the computer model; effectiveness of conducting troubleshooting training on the referent vs. on the computer model.

The transformation process and analytical perspective needed for the creation and development of a computer model from a real world model can be formally characterised in a number of distinct views and phases (Widman & Loparo, 1989) (see Figure 2). Starting from the ''real system'', the source of observable data, the set of limited circumstances, the ''experimental frames'', under which the real system could potentially be observed or manipulated needs to be identified. The model capable of accounting for the behaviour of the real system in all these experimental frames, the ''lumped model'', has to be formulated. The implementation of this model then results in the computer model. Due to this transformation process the computer model is only an approximation of the real world model.

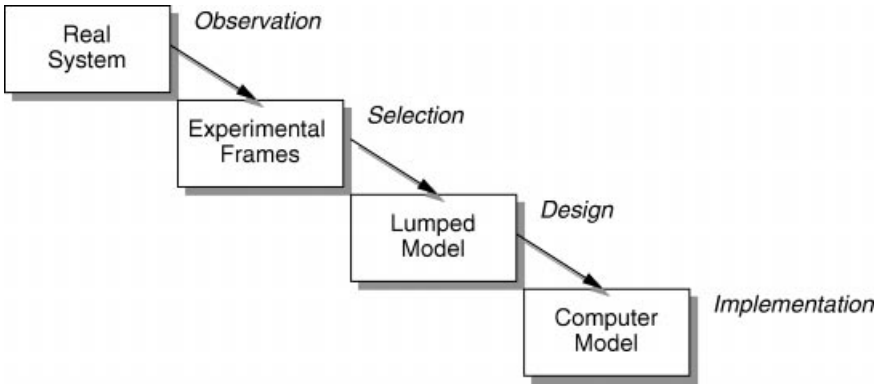

Figure 2 Modelling phases

2.3 Additional opportunities

There are opportunities beyond the ones stemming directly from the virtual nature of the computer-based model. Most of the opportunities are related to the flexibility offered in controlling the computer-based model which can, in comparison to the real equipment, be used to achieve more effective, comprehensive and adaptable training.

Training can for example be made more problem focused by being able to bring the system deliberately into states that are perceived to be important for training. In some instances these states may be emergencies that would normally only be reached in rare occasions in real life situations. Additionally, to enhance learning it is also often useful to be able to redo the training on the model with exactly the same parameters and the same conditions.

The opportunity to deliberately insert faults and failures in the model for troubleshooting purpose enhances the instructional capabilities of the system, and is the precondition for diagnostic training. With the real system, this is often either not possible at all, at least very limited in scope or potentially damaging for the human or system.

The computer model provides opportunity for flexible control of the temporal behaviour of the training system. Very fast executing processes such as for example pressures being propagated through the pipes of a system can be slowed down or stepped through for better observation of effects. Lengthy effects can also be sped up by increasing the simulation speed.

In contrast to the real equipment, the model also enables flexible visualisation. With the computer-based model multiple views on the same model are possible. These can be exploited to select the appropriate level of detail for instruction. It can also provide different perspectives on details normally not observable with the real equipment.

2.4 Overview of systems and application areas

Before embarking on the detailed framework based classification of the approaches, we give a quick overview of the systems to be examined and their areas of application (where available). This is deemed useful for orientation of the reader. It should be noted that only existing and self-contained systems are listed and described in the following. The examples in the subsequent sections refer directly to the design approaches, methodologies and paradigms chosen by one or more of the following systems.

2.4.1 SOPHisticated Instructional Environment (SOPHIE) I, II & III

The SOPHIE system (Brown et al., 1982) provides a ''reactive'' learning environment in the domain of electronic troubleshooting that encourages the explicit development of hypothesis by the student during problem solving and enabling the system to verify and critique them. The three versions of SOPHIE differ mainly in terms of the reasoning capabilities of the system, the coaching strategies

and the student model. Its evolution illustrates the limitation of quantitative simulation for certain instructional purposes.

2.4.2 STEAMER

The STEAMER system (Hollan et al., 1984) ties an elaborate graphical user interface with direct manipulation capabilities to an existing numerical simulation of a marine propulsion system. The user interface provides a qualitative rather than purely quantitative visualisation of the numerical (quantitative) simulation. The system proves the importance of graphical abstractions in the process of understanding complex systems. It has been referred to as a ''discovery world'' for students and has no capabilities of delivering directed instruction to students.

2.4.3 Recovery Boiler Tutor (RBT)

The RBT system (Woolf et al., 1986) is capable of tutoring operators to control a complex system (a recovery boiler) and to solve difficult ''real-time'' emergencies. While the simulation of the system is running, the operator can view the boiler from many directions, such as for example the fire bed. Assistance is provided through visual, acoustic and textual clues. The simulation is interactive and inspectable. The operator can request up to 30 process parameters on the complete panel board and change 20 setpoints. All variables are updated in real time.

2.4.4 Intelligent Maintenance Training System (IMTS)

The IMTS (Towne & Munro, 1988) addresses the question of domain modelling with the behavioural object approach where the domain model consists of interacting objects each of which has its own graphical appearance and is capable of changing it to reflect the actual state of the object within the system model. The IMTS provides concepts and tools for authoring interactive graphical simulations, but the model authoring approach is limited as the system supports no other form of delivering directed instruction to the student than troubleshooting assistance.

2.4.5 RAPIDS

The RAPIDS system (Towne & Munro, 1992), is a follow-up of the IMTS and pulls away from the strictly object-based interaction concept of the IMTS. The RAPIDS system is even more designed as a comprehensive domain authoring and delivery suite and for example provides direct manipulation authoring techniques for instructional content. The system has been used to produce several simulations mostly in the military domain. The system reveals the shortcomings of the IMTS system, for example in terms of the variety of instructional content that can be developed and delivered.

2.4.6 CyclePad

The CyclePad system (Forbus & Whally, 1994) is used to help engineering undergraduates understand important principles of thermodynamics. It provides a conceptual CAD environment where students can design and analyse power plants, refrigerators and other thermodynamic cycles. CyclePad uses and combines several existing AI techniques such as compositional modelling to represent and reason with modelling assumptions, qualitative representations to express the intuitive knowledge of physics to detect impossible designs, truth maintenance to provide the basis for explanations and constraint reasoning and propagation to provide efficient mathematical reasoning.

2.4.7 Turbinia Vyasa

The Turbinia Vyasa system (Vasandani & Govindaraj, 1995) trains operators to troubleshoot faults in a marine power plant. It focuses on balancing qualitative and quantitative modelling to model large complex systems for simulation-based training. To achieve this, the system employs a new modelling methodology (''qualitative approximation'') and its hierarchical model structuring.

2.4.8 Rapid Intelligent Tutoring System Development Shell (RIDES)

The RIDES system (Munro et al., 1996) is an integrated simulation-based courseware authoring and delivery environment. Although it is based on the behaviourally defined object-model approach, the focus is more on a graphically flexible authoring environment and instructional content editing functions than on the object-modelling. Several varieties of simulations have been created with the RIDES system, for example a B2 aircraft landing-gear troubleshooting tutor and an air traffic control operations tutor (Fleming & Horwitz, 1996).

2.4.9 VRIDES

The VRIDES system (Johnson & Rickel, 1996) is an extension of the RIDES system into VR. The RIDES system is used as the underlying simulation driver, but the visualisation of the simulation objects is done through a strictly separated VR package. The system fully decouples the RIDES-based object model from the visualisation (in VR). VRIDES also introduces the concepts of agents into simulation-based training environments. One application of the VRIDES system is to teach students how to operate components of turbine propulsion systems (aboard US Navy ships) in a virtual 3D environment.

2.4.10 Adaptive Visual Simulation Environment (ADVISE)

The ADVISE system (Helfesrieder & Shankararaman, 1998) focuses on creating views on object-based simulations. To achieve this, it formalises and enables the authoring and delivery of simulations into the stages of object-based model creation, data enrichment through characterisation of model objects and high-level definitions of views that are later dynamically assembled at runtime using the enriched model data. The system can be applied to create multiple views on the same model for enhanced training. An initial scenario models parts of an automobile engine and instructionally useful view-definitions on top of the model.

3 A framework for classification of approaches

A review of the applications in the field of model-based simulations for training reveals the diversity of the approaches. However, there is a lack of a comprehensive and commonly accepted methodology or framework for designing the components of the model-based simulations for training. This is true for both the instructional as well as for the simulation component and foremost for the aspects of integration between the two.

3.1 Defining the framework

Many of the approaches are driven by different intentions and emphasis, often trying to prove a single scientific fact or solve a specific problem. There are almost no building blocks available, which could be used as a common base for model-based simulation development.

This makes the direct comparison of existing systems very difficult. As always, when common ground is lacking, trying to directly compare each application with another is not fruitful and a direct comparison with a strict classification scheme would also probably yield incorrect results. Therefore, we identify a number of features that enable a structured investigation of model-based simulation systems for training. These features include:

- modelling methodology: this refers to the basic modelling approach that has been taken to construct a model based on which training can be conducted. The two ends of the continuum are quantitative and qualitative modelling;
- simulator integration: this refers to how deep or shallow the model-based simulation is integrated with the rest of the system, particularly the instructional component of the system. It is an indicator of how the instructional component can interact with the model-based simulation and exploit the simulation for its use;

- model visualisation and inspection: this refers to the human–computer interaction aspect of the system. It relates to the type of interaction offered for viewing or inspecting the model in an exploratory manner;
- model fidelity: this refers to the balance between the model-based simulation and the level of realism necessary to provide effective training and the necessary abstractions that the model has to make compared to its referent;
- support for diagnostic and troubleshooting training: this refers to the capability for modelling abnormal behaviour. The key issues are the support for introducing faults, controlling the effects of the faults and simulating realistic behaviour of the faulty model;
- object-oriented approach: this refers to the extent of use of object-oriented approaches for developing and maintaining the model;
- model authoring: this refers to the approach taken to support the creation of a model for model-based simulation and subsequent refinements or maintenance.

Based on the above features, the following subsections provide a detailed overview on the various approaches to developing model-based simulation for training in ICAI.

3.2 Basic modelling methodology

The two ends of the modelling continuum are quantitative and qualitative modelling.[1] Positioned on that continuum are several different approaches to modelling. There is not one optimal approach to modelling, every approach has its advantages and disadvantages.

3.2.1 Quantitative and qualitative modelling in ICAI systems

Most of the simulation-based ICAI training systems are of a hybrid nature as they mix qualitative and quantitative representations. If pure quantitative modelling assumptions or empirical decisions in the model are not made explicit and not easily verifiable and may not be valid for certain classes of inputs or when faults are introduced (Wenger, 1987). It is, therefore, best suited to producing accurate simulation of a system but unable to clearly reason and explain the underlying model to trainee, therefore, is not very effective for training.

Simulations based on pure qualitative modelling inherently offer an interpretation of their state, since the methodology is built on explicit state transitions, explicit parameter relations and explicit representation of quantities in context-specific quantity spaces. The problem, often referred to as ''branching'', with this approach emerges when multiple state transitions are possible based on the current state of the model (de Kleer & Brown, 1983). When ''branching'' occurs it is impossible to infer the single correct behaviour of the model. In an instructional setting this is not acceptable (Widman & Loparo, 1989).

To avoid the above problems, many systems integrate parts or adaptations from both methodologies. A simulation, which has a purely numerical core, could for example have a qualitative and interpretative shell around this core to reason about the core simulation, thus transferring the quantitative measures to qualitative ones and therefore making them useful for instruction, for example as in the approach taken by the SOPHIE system (see Brown et al., 1982).

Branching in purely qualitative simulations can be partially avoided for example by encoding additional information about branching decisions in an external knowledge base. Other approaches deal with this problem by deriving additional constraints from the structure of the model or by increasing the number of discrete variables in the quantity space and by ordering their influences within the model (Widman & Loparo, 1989). Others have suggested methods such as employing heuristics for decision-making (de Kleer & Brown, 1983).

[1]For a general overview of approaches to qualitative modelling see, for example, Schut and Bredeweg (1996).

3.2.2 Qualitative Approximation

Qualitative Approximation (QA) has been introduced as the modelling methodology for a marine power plant training simulator (Vasandani & Govindaraj, 1995; Govindaraj, 1987). QA is a modelling approach capable of addressing the issues of system complexity such as system size and subsystem interactions. The methodology provides a balance between quantitatively centred modelling with very few state information available on the process and the qualitatively centred approach with very few exact and detailed information available to successfully model a large and complex system. To achieve this, QA relies on maintaining a simplified hierarchical model of a complex system by aggregation, decoupling and order-reduction of its subsystems. System states are represented qualitatively (such as ''pressure high'' or ''flow rate has been steadily decreasing'') and are derived by quantising quantitative values at a primitive level which is the lowest level in the model. For example, QA sets up a hierarchy of the system by decomposing the model of the power plant into four levels of abstraction. On the top level the basic steam power plant is decomposed into subsystems (e.g., ''Condensate and Feedwater'' subsystem). The subsystems themselves consist of components (e.g., ''Condenser'' component) which are assembled of one or more primitives (e.g., the ''Condenser'' is built from the ''Heat Exchanger'' primitive).

This methodology reportedly has problems with system stability originating from the use of numerical representation and the discrete time interval updates. The problem mainly arises with components in feedback-loops and in components directly connected to a failed component. In implementations this may be countered by reducing the numerical output of primitives by a constant factor or resetting selected state values with each step in the simulation.

3.2.3 Object-based behavioural modelling

Object-based behavioural modelling has been used to model a number of simulation-based training scenarios including highly complex simulations (Towne and Munro, 1988; Towne, 1987) such as a helicopter bladefolding system as well as simulations of moderate complexity. In this methodology, a model based on the local behavioural definitions of its basic objects is constructed. These behavioural definitions are not limited to graphical appearance (sometimes also referred to as ''physical appearance'') and can contribute to

- providing a graphical representation of the behaviour and simultaneously run the underlying behavioural model, and
- running the underlying behavioural model without providing the visual representation of the object.

Thus enabling modelling of ''deep'' model simulations where components which are not graphically visible also contribute to the realistic overall behaviour of the system.

In object-based behavioural modelling, internal behaviour rules (condition-action pairs) specify the behaviour of the object with respect to the input it receives to produce the corresponding outputs to other objects. Although the rules reside in separate objects, the overall collection of rules can be thought of as a form of knowledge-based system, however a ''distributed'' one (Towne, 1995).

Typically, the dynamic behaviour of the model is achieved by an inference engine processing all the objects affected by a change through evaluating their behavioural rules. This change can refer to a user triggered action (e.g., turning on a switch) and is sufficient to achieve a responsive system behaviour. Effects of change other than user triggered actions can also be event based, for example the switch can be turned on after a certain amount of time has elapsed.

The object-based behavioural modelling approach does not necessarily implicate a decision between quantitative and qualitative modelling. In fact, both can be used and mixed as the production rules leave it open whether quantitative calculations or qualitative use of states should be used to determine the behaviour of the object.

The approach is based on separated and fully defined objects with a clear interface definition regarding their inputs and outputs. This emphasises the potential reuse of model parts by allowing

insertion and removal of objects and groups of objects from the model, thus facilitating rapid prototyping.

3.2.4 Qualitatively enhanced mathematical modelling

Mathematical modelling has been used in the Recovery Boiler Tutor (RBT) system (Woolf et al., 1986) to simulate equipment and process flow of a complex boiler system. The simulation consists of a set of mathematical formulae which act upon 35 different process parameters (such as steam pressure, steam flow, steam temperature, etc.).

The system operates in "real time" as it updates its status by recalculating the new system status from the current process parameters in 1 or 2 second intervals. With each simulation step RBT also generates aggregate and abstracted measures (for example, of the thermal, chemical and environmental performance) of the system by compiling several other process parameters on the basis of complex mathematical formulae.

Since a pure mathematical model is not capable of providing qualitative information on the state in which the system is in or the underlying causality or why the system has assumed this particular state, additional knowledge is needed to enhance the mathematical model qualitatively.

RBT uses an additional knowledge base to extract qualitative state information from the model for instructional purpose. The knowledge base is used to specify a variety of emergency situations that can occur while operating the recovery boiler. It also incorporates information on numerical values and selected trend information suitable to decide if the preconditions, postconditions or a satisfactory solution for a special training situation, an emergency, has been met. The additional knowledge is processed continuously during the simulation to check if an emergency condition is arising or an existing emergency has been satisfactorily solved by the operator.

3.2.5 Qualitative physics

A combination of qualitative physics and traditional techniques of AI has been used in the development of CyclePad, a learning environment in which students can design and analyse steady-flow thermodynamic cycles in steady-state mode (Forbus, 1997; Forbus & Whalley, 1994).

Qualitative physics is a modelling methodology that provides the means to formulate and reason about processes and dependencies in the physical world in a qualitative manner. In contrast to classical, quantitative physics, which provides no causal information about changes and processes within a modelled system, qualitative physics carries the causal information that allow to reason qualitatively about it, for example in terms of predictions, explanations or inferences (Wenger, 1987).

Qualitative physics is used in CyclePad to incorporate knowledge about what is physically possible in the design of a thermodynamics cycle. With this knowledge of the physical constraints of components and the processes inside each component, it is possible to detect impossible designs. To achieve this, the ordinal constraints included in the process occurrences inside each component are tested against numerical values defining the properties of a component. This is accomplished by a constraint propagation mechanism included in CyclePad. CyclePad is one of the early systems to demonstrate the applicability of qualitative physics in a comprehensive instructional and educational setting.

3.3 Simulator integration

For reasons of modularity and efficiency simulators in simulation-based training systems are usually separated from the instructional components of the training system, which is also reflected in the system architecture. In practice this means that the simulator is to a certain degree opaque to the instructional system, and does not offer ready access to its inner workings and data-structures.

Every simulation-based training system must address the question of integrating or coupling the simulator with the instructional part of the system, since both parts contain information crucial for the successful operation of each other. The degree of coupling that can be achieved can be divided into three levels: no coupling, shallow coupling and deep coupling.

3.3.1 No coupling—Stand alone simulators in ICAI

No coupling may be the case when there is no instructional component available and the standalone simulator may or may not provide some instructional functions. In this case the simulator alone has limited training capabilities. The STEAMER system (Hollan et al., 1984) serves as an example for this, although it should be emphasised that the main intention of STEAMER was not to produce a comprehensive instructional system but to demonstrate learning about a complex system through concepts of graphic abstractions and direct manipulation.

STEAMER is a system without an explicit tutorial component and does not teach a specific task. Though a student in STEAMER can be asked to perform a specific procedure in successive steps, its instructional capabilities do not go far beyond an exploratory discovery learning environment. The simulator alone has no capability of providing additional help based on for example student's misconceptions which would involve at least a basic student model. Furthermore, it can offer no guidance for the trainee within the instructional session because this would require an additional pedagogical component.

3.3.2 Shallow coupling vs. deep coupling

The integration of the instructional component and the simulator can be characterised as being shallow or deep with reference to the degree of access the instructional component has to the simulator, and *vice versa*.

Shallow coupling refers to when the tutoring component treats the model-based simulation as a black box. It has no possibility to verify how results were derived within the simulator. Accuracy, assumptions, limitation and derivation mechanism of the computations within the simulator are opaque in this case. The classic shallow coupling approach has been taken by the first versions of the SOPHIE system.[2] SOPHIE II treats the simulator, which is a general non-AI tool, the SPICE circuit simulator, as a black box model. The assumptions and limitations of the simulator are dealt with the so-called procedural experts (Rickel, 1989; Brown et al., 1982). These procedural experts have knowledge about the simulator in terms of knowing how to run it, what its limitations are, and how to interpret and use the results gained.

In deep coupled systems, some relevant parts of the simulator are made visible to the instructional component. For example the instructional component is able to select the appropriate simulation algorithms, knows about their limitations and may be able to modify the algorithm. For example the Recovery Boiler Tutor System (RBT) is built on an accurate mathematical model. However the emergencies that can occur in the RBT are encoded twofold (Woolf et al., 1986):

- each emergency is held in a knowledge base and described in a frame-like structure with slots for preconditions, optimal actions, and conditions for solution satisfaction. This knowledge base enables the system to decide which of the mathematical formulae to apply for a specific emergency;
- the emergencies in the steam boiler are represented as a set of mathematical formulae which ensures that the process parameters can be produced accurately in the simulation.

In this way, the RBT can adapt to a variety of special situations, the emergencies in this case, by temporarily extending the model with the appropriate additional mathematical formulae which ensures accurate simulation of the model in a particular situation.

3.4 Model visualisation and interaction

In a simulation-based training system model visualisation and interaction refer to the Human Computer Interaction (HCI) approach adopted by the system. It characterises the role of the

[2]Actually three SOPHIE systems exist, each of which has a different main focus. For us, the main interest lies in the qualitative understanding of the circuit simulator as in SOPHIE III and the drawbacks in using an opaque simulator as in SOPHIE I & II.

interface and the two interacting entities, trainee and computer-based training system respectively, during the interaction.

For better investigation within the context of simulation-based training systems, we break up the interaction into the aspects of visualisation, inspection and manipulation of the model.

- Visualisation refers to the graphical presentation of the computer-based model and the related simulation process, possibly from multiple view points and with various visual enhancements.
- Inspection refers to the interactive character of the transactions taking place between trainee and the model. It characterises the extent and methods with which the computer-based model allows and supports observation and navigation by the trainee.
- Manipulation refers to the ability of a system to allow, in most cases, direct manipulation and control over the simulation through access to a number of parameters of the simulation. In contrast to inspection, the trainee is able to modify the inner working of the simulation.

3.4.1 Enhanced model visualisation

The capabilities of graphical representation of the computer-based model of a complex system go far beyond those of the referent. In contrast to the referent the knowledge underlying the computer-based model is represented in forms of structural, functional or behavioural information. This computer-based representation may be expanded or reduced as needed. For the purpose of training, it may be exploited to generate more focused and context-adapted representations of the model.

Multiple perspectives. Creating different views on a computer-based model allows a variety of qualitatively different points of view to be investigated. This can refer to revealing hidden objects that are not observable in the referent, viewing the model on different levels in an abstraction hierarchy, presenting the model at different levels of detail or viewing the model from different spatial perspectives in 3D representation.

Hidden objects: The IMTS Bladefold system (Towne, 1987), which trains operators on aspects of the bladefolding operation of a helicopter, is capable of depicting interaction between various objects that would normally be invisible to the observer on the actual equipment. System components that are normally hidden by opaque material exhibit their states and interactions with other components. The trainee can observe valves opening and closing or cylinders extend or retract. With this representation of normally hidden objects the model-based simulation is able to reveal aspects of the Bladefold system which would be difficult to discern using the real equipment.

Different levels of abstraction: A complex dynamic system can be perceived in an "abstraction hierarchy" (Rasmussen, 1985), where the top levels are concerned with abstract high-level aspects and the lower levels with the more detailed physical form of the system and its components. Providing graphical representations of a model that reflect this hierarchy is of instructional benefit because every level can contribute specific conceptual, structural and informational aspects of the semantics of the whole system (Rasmussen, 1985).

The STEAMER (Hollan et al., 1984) system has emphasised the importance of hierarchical abstractions and forms for their graphical representation. It is able to depict the steam plant on several levels of abstraction. About 100 different views have been created to cover the steam plant on various levels of abstraction. The top-level view in the abstraction hierarchy is the basic steam cycle of the steam plant. It shows the whole steam plant with the important main components. This viewpoint serves as the source of high-level information about the overall plant topology, causal structure and connectivity of the steam plant.

Additionally, STEAMER allows to focus on the subsystems that make up the steam plant. A subsystem view depicts the causal structure, topology and the dynamics of a specific subsystem in detail and provides an abstraction of the specific contribution of the subsystem to the overall functioning of the steam plant. A view of the basic elements on component level is also provided. The trainee can explore structure and functioning of specific components in a "minilab" (Wenger, 1987), that allows assembling and testing of basic system components.

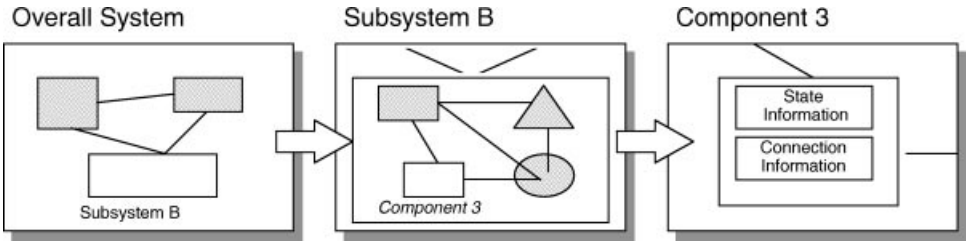

Figure 3 Different levels of detail (top-down progression)

Level of detail: A computer-based model can be visualised in varying levels of detail. Variations in detail can be of instructional benefit and allow, for example, top-down or bottom-up progressions. A top-down progression would initially present a complete, but not very detailed version of a complex system (Towne, 1995). The following progressions would then focus for example on a specific subsystem of the overall system and then incrementally focus on further details of the subsystem (see Figure 3).

In behavioural object model systems such as IMTS or RAPIDS, it is possible to present the computer-based model in different levels of complexity with regard to the visual representation of objects and their interactions. This can be achieved by removing selected graphical objects from the visual representation but retaining the full computer-based model underlying the graphical representation. This results in an unchanged model-based simulation with a reduced graphical representation.

For example, in a RAPIDS-based training scenario which teaches maintenance aspects of a T-38 dual-engine jet aircraft, the schematic representation of the engine-starting components are represented in five different versions with increasing level of detail (Merill et al., 1992). The least detailed schematic presents "T-38 AC Power Generation" which represents only a small portion of the original detailed schematic whereas the most detailed "On-Ground Start" schematic represents all portions. During instruction the five schematics are presented to the learner in the order of increasing complexity.

The ADVISE system (Helfesrieder & Shankararaman, 1998), which is also based on the behavioural object model approach, uses methods of data-enrichment of the simulation model by characterisation, typing and grouping of model components at the authoring stage. With the additional meta-knowledge about a model component being part of a specific hierarchy or semantic context within the model, the system is able to generate different views on the overall model-based simulation depending on the contents of a descriptive query fed into the view-generator module of the system. In an ADVISE-based scenario that models portions of an automobile engine a query may be specified that brings up a view containing model components involved in the context of "combustion" and are situated in the subsystem "combustion chamber". With this methodology, the system formalises and implements a layer of knowledge about the model and its components which allows higher-level specification of views on a model-based simulation.

Use of virtual reality: Graphical representations of a model-based simulation scenario for training are not limited to 2D presentations and can also be extended into 3D space. VR systems rely on a number of enabling technologies that involve specialised hardware and software. These include generation of real time 3D computer graphics, stereoscopic displays, head and gesture tracking, haptic feedback and inclusion of sound, many of which are still subject to further research (Gigante, 1993).

Simulation-based training in a VR environment can: add spatial information for training (e.g., where the specific components and subsystems of the system are located in a control room); realistically represent each component according to its spatial properties; allow focusing on graphical details of a component by "moving" towards it in the 3D space and support more natural and realistic interaction with the environment through its graphical representation and visualisation.

The VRIDES system (Johnson & Rickel, 1996) is a VR-based ICAI system in which the model of a complex dynamic system, in this case a part of a gas turbine propulsion system (used on US Navy ships), can be observed and operated by the trainee. VRIDES generates the VR environment by combining an extension of the 2D model-based simulator RIDES (Munro & Pizzini, 1996; Munro et al., 1996; Towne, 1995; Munro, 1993) with Vista, a virtual environment system that enables the creation of immersive virtual environments.

As in all immersive virtual environments, in VRIDES the trainee becomes part of the process and data compared to observing it on a 2D screen. The trainee can move around in the scene, for example in a control room setting, and operate buttons and handles of the system or observe indicator lights.

In this approach, an underlying behavioural object-based simulation is continuously run to generate the dynamic behaviour of the modelled system. The resulting visual object behaviours and effects are then presented in the VR environment. The system allows not only to observe the system workings in VR, but also supports various interactions with the VR environment. Additionally VRIDES provides coaching and assistance in the form of a synthetic agent in the same VR environment (Johnson et al., 1997).

Explicit parameter visualisation. Explicit visualisation of specific parameters of the model over a period of time can also be of instructional value. These parameters are in many cases "critical" variables, abstract measures, or reflect the relationships between domain parameters.

Parameters can be presented in the form of trends where the change of the parameter during simulation is shown over a time frame. Trends support assessment of the current process situation and prediction of the future development of the specific parameter.

Composite measures combine single parameters to generate more general measures reflecting abstract information about the state of the model. Composite measures contribute to the instructional value of the model-based simulation. In the Recovery Boiler Tutor system composite measures such as the overall boiler "safety" are calculated from selected basic process parameters and presented to the trainee to support reasoning about the complex process (Woolf et al., 1986).

Model-based simulations can also make use of the first derivative of a parameter to obtain the direction and rate of change of parameters at a specific instant. The first derivative has been used in STEAMER to obtain information whether the signal (indicating pressures or air-flows, etc.) is currently rising, falling or steady (Hollan et al., 1984). This information is in many training situations more relevant than the exact current level of the signal because it relieves the operator from comparing exact numerical values and supports the qualitative assessment of a scenario. The first derivative can be seen as a form of qualitative view on a quantitative process and the authors of STEAMER have coined the term "continuous graphical explanation" to characterise this.

Representing invisible processes. The opaqueness of the referent can hinder observation and reasoning about processes taking place within the system. Additionally, the nature of the involved substances can also cause problems. For example, pneumatic pressures, liquid flows or electricity are often difficult or impossible to observe within the referent. Graphical representations of a computer-based model enable observation and reasoning about normally invisible processes. Flows can, for example, be made visible by depicting them with moving lines in a seemingly transparent pipe where fast moving lines indicate a fast flow and vice versa (Towne, 1995). Additional aspects such as perception and graphical representation of displaying flows and motion are discussed in Bennett (1992).

Completely invisible processes such as the flow of electricity in an electronic circuit can also be made visible. For example, the clock pulses of a signal going into a chip can be visualised as a train of square-waves moving into the chip's clock input at a speed slow enough to allow observation of the behaviour of the chip regarding the outputs it produces (Towne, 1995).

3.4.2 Model inspection and direct manipulation

While aspects of representation and thereby observation and observability (as discussed above) is an

important part of training and instruction, active participation is another key element in a training and instruction setting. The trainee must be given the opportunity to apply the previously acquired knowledge.

To support active participation, the model-based simulation has to be inspectable and manipulable by the trainee.

- Inspectability refers to the flexibility offered by the model to allow the trainee to inspect its internal workings.
- Direct manipulation is an interaction technique which emphasises manipulation of variables of the model-based simulation directly on screen and presenting resulting effects through immediate visual feedback.

The term inspectability covers a variety of aspects since it describes, how observable, accessible and controllable the model-based simulation is for the trainee. It covers aspects such as the non-opaqueness of material within the model, display of parameters even if no actual instrumentation exists, support for stopping the simulation or single stepping through procedures to gain detailed understanding of the effects taking place. Also the opportunity to create hypothetical situations by deliberately altering specific parameters or by inserting faults into the model contributes to the inspectability of the model-based simulation.

Direct manipulation (Hutchins et al., 1986) in contrast is a technique (as such independent of the area of application) which enables interaction and, therefore, contributes to the instructional value of training. Direct manipulation usually takes place through a "first person" interface (Laurel, 1986) and allows direct manipulation of graphical objects in the interface.[3] In STEAMER (Hollan et al., 1984), the trainee can directly manipulate devices. For example, a valve can be closed by clicking with the mouse on it rather than by issuing a command like "CLOSE VALVE 17X" (Miller, 1988). The paradigm of direct manipulation can be beneficially combined with an inspectable and "reactive" model-based simulation to form the basic building blocks of an explorative instructional environment.

A number of potential explorations such as allowing the trainee to manipulate domain variables to perform experiments or to change system displays in order to provide additional decision support and alternative conceptual perspectives in complex dynamic systems are discussed in more detail by Bennett (1992).

3.5 Model-based simulation fidelity

Simulation fidelity refers to how closely the simulation imitates reality. Research has shown, that although fidelity is a critical variable in instructional simulations, this does not imply that high-fidelity is crucial (Alessi, 1988). Instead, the simulation fidelity can be varied to provide optimal instruction and transfer (Shankararaman & MacFarlane, 1992).

The fidelity requirements of the simulation-based training system are always related to the instructional objectives. If for example interaction with simulated equipment is a major portion of the operational task to be instructed, this fact will be reflected in the fidelity specifications of the simulation (Hays & Singer, 1989).

Aspects of simulator fidelity combined with simulator effectiveness are usually discussed in terms of metrics and fidelity features. Different metrics have been proposed (see, for example, Baron, 1987) to translate research and development specifications into simulator design specifications. They, for example, measure the degree of correspondence between the physical and the simulated training environment, measure the stimulus environment of the trainee, classify the types of errors committed on the part of trainee or to measure fidelity in terms of the stress and workload.

[3]In contrast to this, in second person interfaces the trainee passes a command to some form of intermediate instance which then carries out (possible an own interpretation of) the command (à la RBT or SOPHIE).

In the following we will look at a possible metric suitable to describe the fidelity of a domain simulator component within an ICAI system for training.

3.5.1 Fidelity metrics

Simulator fidelity is often interpreted as ''greater physical realism'', which is questionable, at least regarding model-based simulation for training. For a training objective such as orientation or identification of system parts, a higher degree of physical realism may be useful. For a training objective such as to understand how the pressure in a water tank relates to a particular valve within a system, a high degree of realism is irrelevant or may even be counterproductive because the abstractions that need to be beneficially employed may not necessarily reflect the physical reality of the real system.

In ICAI model-based simulations for training the paramount training objective is usually to convey knowledge and skills of the training domain to the trainee. This is usually related to tasks such as understanding operation and workings of the system by observation, actively practising, identifying parts or subsystems and troubleshooting a faulted system.

These kinds of training objectives require certain amounts of simulator fidelity, however, balanced in different ways. A number of fidelity features are suitable for describing the overall fidelity for an ICAI simulator for training. These include structural fidelity which measures the structural detail, completeness and connectivity of its components and subsystem hierarchies; dynamic fidelity which refers to the accuracy and dynamic comprehensives (variety of system states) with which the simulation unfolds over time; temporal fidelity in this context as a subset of dynamic fidelity and has been used to describe either the accuracy of simulation timing (Towne, 1995) or the accuracy of sequencing of evolving simulation states (Govindaraj, 1987); and physical fidelity which refers to the degree of physical correspondence between the modelled system including related ambient conditions and the representation provided by the simulator. In addition to these objective measures, the subjective perspective (''perceptual fidelity'') of the trainee must be considered.

3.5.2 Structural fidelity

Structural fidelity is a measure for the structural detail, completeness and connectivity with which components and subsystem hierarchies of the simulation model are represented. It can refer to the level at which single components are still accurately modelled or from which components of the real system have been aggregated. Additionally, not only structural details can be omitted in the modelling process, but complete subsystems can be ignored if they are deemed irrelevant in the training scenario. The detail with which interconnections of components are modelled influences the level of structural fidelity. A structurally high fidelity simulation would model and describe the interconnections between components as complete as possible. For example, interconnections could be classified according to the load they carry. The system topology in terms of where components are located or how pipes or wires interconnect them is also an important aspect of structural fidelity.

Training that takes place on the level of system components and their interactions, such as for example diagnostic training, requires a high level of structural fidelity. To practice model-based troubleshooting the trainee needs to understand how faults on a structural level (e.g., interconnected components) change causal relation and thereby behaviour of the simulated system. Additionally, as troubleshooting in complex systems involves investigation of the system on multiple hierarchical levels of abstraction (Rasmussen, 1985), an adequate level of hierarchical structural fidelity is necessary as well.

3.5.3 Dynamic fidelity

Dynamic fidelity refers to the granularity, comprehensiveness and accuracy with which the simulation evolves. It compares the overall state space of the model-based simulation with those the real system can possibly assume. Although it includes the measure of accuracy it rather refers to a coarser overall completeness of the simulation and how the modelled system is covered in its

behaviour as a whole. More detailed fidelity features such as accuracy of timing or sequencing can be described with the measure of temporal fidelity.

A complex system in an instructional scenario can be simulated with varying detail. On the two extreme ends it can be modelled only capturing the very essential process variables and dependencies or it can be modelled with a high level of detail including "fringe" process details as well as the essential ones. From an instructional viewpoint it should be noted that it is not necessarily a drawback to work with a rather reduced dynamic fidelity domain simulation since it can focus the trainee's attention on the important concepts of the domain. It may, for example, be useful to reduce the simulation to a number of important emergencies which can occur in the real system. However, a low dynamic fidelity influences the instructional effectiveness of the simulation in terms of instructional comprehensives. If the simulation does not cover all possible states of the modelled system, the training on the system cannot be entirely complete.

Assuring the accuracy of the evolving system states is especially important in complex systems where the high level system behaviour is produced from low-level interactions of small system entities. The high level system behaviour may often not be directly obvious from the low-level system structure and reveals itself not until the simulation is actually run. For example, this is relevant in systems modelled with the behavioural object approach. In these systems, therefore, a heavy burden is placed on the model-construction tools and methodologies (e.g., tools for iterative model debugging are needed) and on the expertise of the model authors.

A simulation of a complex system that evolves to wrong system states compared to the real equipment can be regarded as being potentially misleading and instructionally unsound.

3.5.4 Temporal fidelity

Trainee performance in connection with workload and timing can be important factors in training of the operation of complex systems. In certain real world operational settings the environment might significantly change while the trainee is thinking about his or her next action. In these settings it is vital to train or assess co-ordination and timing (Shankararaman & MacFarlane, 1992). The RBT system, that instructs a trainee in the operation and control of a complex industrial process, ensures temporal fidelity by updating its dynamic process meters in fixed seconds intervals. Although the system allows "stopping" the processes for pedagogical reasons, the system is able to provide a somewhat "real-time" training environment.

However, partially neglecting temporal fidelity in terms of accurate timing can be of instructional value. Processes that are too fast to be observed in detail, such as propagation of pressures through a complex system, can be slowed down or even single stepped through (Towne, 1995; Hollan et al., 1984). Other processes that occur over a long timespan, such as the travelling of planets through the orbit, are to be sped up to allow effective observation of the process.

In some instructional settings, especially diagnostic training, the sequence of system state evolution bears important information for diagnostic observation. The sequence of propagation of fault effects over time usually plays an important role for identification of the problem category and the origin of the fault. In such scenarios where diagnosis relies on rather qualitative reasoning about the observed system, exact state values and with that an exact timing of state changes is not important. What is most important is a high temporal fidelity in terms of the accurate sequence in which the state changes occur since the sequence in many cases indicates the causality underlying the fault in the complex system.

In object-based simulation system temporal fidelity is needed on component level where the order of component state evaluation and updating is important. Problems of evaluation and update ordering are usually associated with aspects of serialisation and discretisation of an object-based simulation. When for example a pipe component is connected to a "tee" that leads into two subsystems which then again come together, the question arises which one of the subsystems should be evaluated first (Towne & Munro, 1988). If the evaluation of one branch has an influence on what happens in the other branch, the simulation can yield incorrect results.

3.5.5 *Physical fidelity*

Physical fidelity refers to the degree of physical correspondence between the modelled system including related ambient conditions and the representation provided by the simulator. Physical fidelity includes aspects such as the degree of correspondence of visual appearance of the modelled system, how realistic the system feels in the sense of touching surfaces and textures, but also kinaesthetic aspects such as force feedback during interaction with the environment such as pushing, pulling or pressing. Also ambient acoustic or smell (for example smoke can indicate danger in a training environment) information contribute to physical fidelity.

It is obvious that some of the features such as smell are very difficult to handle for today's simulation technology. Aspects of touch and kinaesthesis require physical and spatial interaction with the environment generated by the simulator. These requirements can be partially matched with the VR technology and methodology. Immersive VR provides a spatial representation of the environment with which the trainee can interact. The sense of feeling surfaces and textures of objects as well as the "sense of muscular feel" can be provided with force feedback mechanisms.

One of the essential advantages of immersive VR in training is the natural and intuitive interaction within the virtual world. In this virtual reality the cognitive load of the trainee is reduced because scenario and objects conform to the user's natural expectations, thus freeing resources for the actual training task (Earnshaw et al., 1993).

The VRIDES training system enables trainees to perform training tasks in an immersive VR environment (Johnson et al., 1997). The trainee observes, inspects and interacts (by directly manipulating) with spatially defined objects such as pressing buttons, pulling levers or reading indicators. Additionally, the trainee can also move around in the training scenario in a natural manner.

3.5.6 *Perceptual fidelity*

The above metric is suitable for describing the objective appearance, behaviour and structuring of a model-based training simulator. From the viewpoint of the trainee, fidelity can be described subjectively in terms of what the trainee perceives when interacting with the simulator (perceptual fidelity). This measure takes into account the capabilities and limitations of a human trainee. Perceptual fidelity is especially important in ICAI simulation-based training scenarios since they are designed for effective knowledge transfer and thereby centred around the trainee. This gives room for partial neglection of some of the above stated measures of objective fidelity.

3.6 *Modelling abnormal behaviour*

As an important precondition of diagnostic training, the trainee needs a sound and profound knowledge of the complex system and its operation under abnormal conditions. Experience shows that learning how to diagnose and repair a complex system is acquired through practice (Towne, 1987). Practising in the sense of applying knowledge is crucial in diagnostic training. Moreover, practising troubleshooting can also affirm the deep understanding of the functioning of the complex system.

3.6.1 *Faults in complex systems*

Components in complex systems can exhibit failures in many different ways and affect other components in the system. This can cause other system components to fail or to otherwise exhibit abnormal behaviour. Furthermore, the effect of faulty behaviour of a component is usually not limited to abnormal behaviour at the component level but can also cause abnormal behaviour in high level subsystems or the overall system. This potentially far-reaching characteristic of component failures is symptomatic of the dynamic and complex nature of system faults. Additional complexity is introduced by the fact that not only functional components in the system can be faulty but various meters and displays can malfunction as well, indicating faulty behaviour where the system is in fact functioning properly.

Failures in complex systems usually at least lead to a degrading overall performance of the system, or can result in dangerous situations and system states that potentially endanger man and machine. This makes it even more worthwhile to train operators of complex systems in troubleshooting and system diagnostics.

3.6.2 Impact of faults on modelling

Most model-based simulation systems that train the operation of complex systems engage to some degree in diagnostic training. Some almost exclusively focus on troubleshooting training (such as Turbinia Vyasa or RBT) while others treat it as only one of a number of instructional goals.

Failures have a potentially high dynamic and complex impact on the system, therefore the modelling of faults is difficult. One must ensure that the faulty model displays a realistic behaviour under the influence of the fault (compared to the real system) which may even lead to a continuously changing behaviour as the fault propagates through the system, thereby possibly causing additional failures in other components. Regarding a model-based simulation this means that failures have a potentially great impact on the system's model as they may alter its basic structure, function and behaviour. The basic modelling methodology of the complex system, therefore, imposes limitations on the type, number and quality of failures that can be modelled in a system.

3.6.3 Different approaches to troubleshooting

Due to specific training objectives and rather technically imposed limitations, model-based training applications have taken different approaches to diagnostic training. Past approaches have enabled the trainee to diagnose abnormal conditions on the bases of abstract process measurements (for example, the RBT system), identifying faulty system parts and components through observation (for example, the Turbinia Vyasa system), and by inspecting and interactively troubleshooting complex systems on component level (for example, the IMTS). Consequently, the troubleshooting interactions of the trainee with the model in the diagnostic process are very different, ranging from rather simple normal-abnormal assessment to sophisticated test and repair interactions where the trainee is required to replace suspected components. We have classified these into three main approaches namely, process-based troubleshooting, troubleshooting with qualitative representations and component-based troubleshooting.

Process-based troubleshooting. In the Recovery Boiler Tutor system, the simulation is mainly based on a pure mathematical model. The trainee does not actually troubleshoot or diagnose equipment, but only an abstract production process in a number of emergency scenarios (Woolf et al., 1986). RBT does not maintain knowledge on structure or topology such as subsystems, components and their interactions. Rather than directly inspecting components and their interactions the trainee conducts diagnosis by observing and actively inspecting measures about the process taking place inside the boiler.

The trainee interacts with the process by choosing predefined actions such as "Check instrumentation" or "Start standby feedwater pumps" from a menu. Troubleshooting is successful when a sequence of actions appropriate for the actual emergency is performed by the trainee. Information on sequences for successful actions are stored in an additional knowledge base. In RBT, abnormal behaviour is achieved by executing additional formulae specific for an emergency situation. This ensures that process-measures and readings during an emergency can be accurately reproduced.

Troubleshooting with qualitative representations. In Turbinia Vyasa, a simulation based on the qualitative approximation methodology, selected components fail and affect other components within the system (Vasandani & Govindaraj, 1995; Fath et al., 1990; Govindaraj, 1987). The trainee diagnoses components of a complex system by inspecting specific state information through gauges. Number and location of measuring points are fixed and are arranged according to the real system, thus providing the trainee with the same information and access to the system as he or she would have in the real system but not more. As in the actual system, the gauges reflect only a limited

number of component properties such as pressure, temperature, flow/level and smoke. The diagnostic interaction is successful when the failed component has been identified.

In Turbinia Vyasa, only one component fails at a time. The failure is described in a predefined "failure-frame" that contains details of the current failure such as information on the state-variables of the affected component and information on which other components are affected by the fault. Information from the failure frame must be processed and applied continuously during the simulation to ensure that the system converges to a steady state. This enables the system to have information about the fault and then control its effects on the simulation. The effects of the failure are initially propagated through the system which provides valuable hints for the trouble-shooter. The propagation continues until the system stabilises under the influence of the fault due to continuous resetting of specific component properties according to the failure frame.

Component-based troubleshooting. Behavioural object modelling, as employed in systems like IMTS, RAPIDS or RIDES, concentrates on building models of complex systems from their low-level components. The behaviour of each component contributes to the overall behaviour of the system. A failure in a complex system basically occurs on component level and may or may not subsequently affect other components which then may or may not change their behaviour accordingly. Due to the availability and access to low-level components, system modelling with behaviourally defined objects is very useful for conducting diagnostic training on a detailed and fine grained level.

In such systems, the modelling of faults comes almost at no additional cost and fits naturally into the modelling approach. The behaviour of a component under normal conditions is described by rules that govern its behaviour in terms of the output it produces for a given input and its graphical appearance. Faults can be introduced by specifying the conditions under which a specific component fails as well as its behaviour in failure mode, for example a blow-out failure due to a too high pressure leading to the failure of the pumping system. Cascading fault effects are achieved with no additional effort, since faults are cascaded automatically by simply propagating the output of faulted components to other connected components (Towne, 1993). This approach also allows replacement at the component level (where the sequence of replacement is important to prevent the new component from being blown again right away).

The above flexibility of fault modelling does introduce a problem, as the system is potentially unaware of the effects that one or more faults cause due to the potentially large number of possible system states that are introduced by a fault. However, instructional support and evaluation in fault-diagnosis can only be provided by the system if it is aware of all the effects of a specific fault on the system. In the SH-3H helicopter bladefolding simulation with IMTS it was, therefore, necessary to automatically insert in sequence all possible faults into the replaceable components of the simulation and record their effect for later use during training (Towne, 1993).

3.7 Aspects of object orientation

Object orientation (see, for example, Booch, 1994) is a powerful software engineering approach that allows to map directly a real world view of a problem onto a software implementation of the problem by focusing on the terminologies, resources and abstractions that are present in the application world.

The choice of employing the object-oriented modelling approach depends on the intentions of the system designer and the requirements of the domain. In some cases, it is not fruitful to apply object orientation to system design. If functioning and behaviour of a complex system are sufficiently captured by a set of mathematical formulae which are readily available then there is no need to go through the additional effort of the object-oriented modelling phases. As shown in the Recovery Boiler and STEAMER systems, the set of mathematical formulae can be taken as the groundwork which can then be adapted to gain the flexibility required for the application scenario or it can be used to drive graphical objects attached to visualise the process (Woolf et al., 1986; Hollan et al., 1984).

Object-oriented approaches have been applied on various levels and in various phases of simulation model development. Object-oriented design should not only be seen in connection with software development but as a general design paradigm that can be beneficially applied to many fields where a complex system is constructed from interrelated entities. Object-oriented approaches have been used to define objects graphically and behaviourally in the user-interface for visualisation, for hierarchical structuring and decomposition of complex system models, and to create complete complex system models from behaviourally defined models.

3.7.1 Object orientation in model-based simulation

A model for simulation can be built by using previously defined objects from an object library. This approach, which has been used by some of the systems (e.g., IMTS, RAPIDS or STEAMER) for model-based training, is in line with many of the ideas of object orientation. It reflects the ideas of types and instances, emphasises reuse of previous work as well as modularity and encapsulation.

3.7.2 Graphical user interface objects

The object paradigm can be used in the user interface design and implementation for model-based simulation. Graphical objects, that visualise objects in the simulation domain such as pipes, dials or switches can be created as specific instances of general objects. This notion of generic-specific instance introduces the principle of abstraction in the design process. For example, a pipe that carries a substance always has a flow rate connected with it which can be depicted with arrows. A specific instance of a pipe may, however, carry a particular fluid through it.

Graphical user interface objects only provide a graphical layer which supports interaction and representation of the domain. The semantics and dynamics of the domain is provided and generated from the underlying simulation model. This separation implies the need to interface the graphical objects present in the user interface with the model-based application working in the background. This can be achieved with a mechanism that links selected variables of the simulation to selected graphical objects.

In the STEAMER system, which relies on a mathematical model, graphical user interface objects with a defined graphical behaviour are connected to certain variables of the underlying mathematical model by a process which the authors refer to as ''tapping'' (Hollan et al., 1984). The numerical values are mapped onto a specific graphical behaviour of the component. This usually involves the need to be aware of different types of variables, for example logical or discrete, and then provide appropriate type conversion mechanisms. As the variable is continuously updated by the executing simulation, the graphical appearance of the user-interface object is updated as well. The graphical objects in the user interface do not communicate with each other. That is they have no semantic knowledge. In fact data is provided from the underlying model and objects just represent and behave. The ''driving force'' behind it is still the underlying model. The interface does not carry much semantics of the domain but is capable of providing interaction and a very limited tutorial knowledge (Stevens & Roberts, 1983).

The ADVISE system (Helfesrieder & Shankararaman, 1998) follows the Model-View-Controller (MVC) paradigm (e.g., Booch, 1994), thus implementing a strict separation between representation of a model-component, its graphical representation and interaction (the control portion) capabilities with it. The separation enables the system to generate variations of views (graphical representations) of the same underlying simulation model. Following a query-based description of a specific view, the system selectively picks components for visualisation and instantiates graphical objects which are immediately linked and synchronised with their associated model-objects. When a model component changes its state, the component is flagged as being ''dirty''. This information may then be used by the associated graphical object to determine if the new state of the component implies a new graphical representation or not.

3.7.3 Constructing complex system models from objects

In the previous subsection object orientation was limited to the interaction and graphical

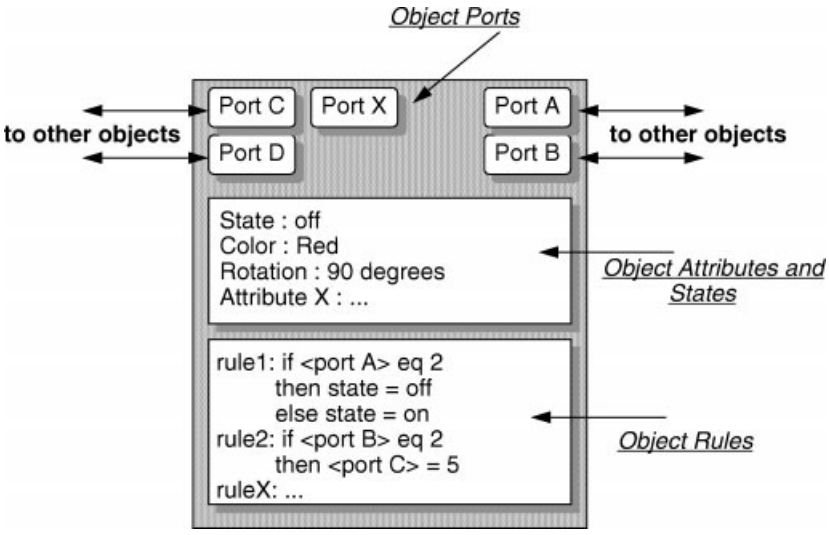

Figure 4 Behaviourally defined object

representation of a model-based domain simulation in an object-oriented manner. The graphical objects were interfaced with the underlying domain simulator but, however, could not be expanded into the model and its semantics.

With models constructed from behaviourally defined objects, the object-oriented design can be fully extended to both the domain model and its graphical representation. An object can at the same time capture its graphical behaviour and appearance as well as its behaviour as a model component (see Figure 4). As all model objects share some basic attributes and functionality, it is useful to formulate a basic layer that is a ''meta model'', which defines a basic framework for connectivity, interaction and behaviour of model objects in a standardised manner. This is needed to ensure the availability of a consistent architecture in which the semantics and dynamics of a model can be expressed. Concrete objects of model components inherit the functionality of this basic layer by building on it.

In object-based systems such as IMTS or RAPIDS (Towne and Munro, 1992; Towne, 1987) the basic structuring of all objects is similar. Common to all objects are ''ports'', enumeration of potential ''component states'', information on ''failure modes'' and ''rules'' defining the behaviour. Connectivity is modelled through the use of ports that allows to formulate interaction between model components in a standardised manner. Ports can be classified as being for example ''hydraulic'' or ''mechanical'' to provide additional connectivity information for the model-based simulation.

Every object within the model relies on these basic mechanisms and functionality, therefore, nothing below this level has to be re-implemented. Each object can concentrate on specifying higher level behaviour on top of this meta-model layer.

3.7.4 Hierarchical decomposition of complex systems

Complex systems can be described in an hierarchical way. In a simple bottom-up approach, system components can be, for example, arranged into subsystems based on their functionality which then constitute the overall system. Such an approach based on decomposition can yield better control over the system's complexity and allow the modelling to concentrate on layers of the complete system.

The object paradigm can support the hierarchical decomposition of a complex system as follows:

- through specialisation on more detailed levels;
- on component level, types of entities can be identified in the application domain from which instances of specific objects can be created;

- objects can be rearranged, for example components within a subsystem, because data and functionality is encapsulated and hidden from the application and other objects.

Hierarchical decomposition of a complex system has been used in Turbinia Vyasa (Vasandani and Govindaraj, 1995), which models a complex system of a marine power plant in hierarchical layers. The top layer which is the marine power plant consists of interacting subsystems such as "Power Generation" or "Steam Condensation". Each subsystem comprises a number of components. Each component is an instantiation of a generic type. A component modelling a feed-pump would be of the generic type "Source"; a condenser component would be of the type "Heat Exchanger". On the lowest level, components can be decomposed into primitives which are responsible for the basic functions performed by that component (Govindaraj, 1987).

3.8 Model authoring

The process of model authoring is concerned with assembling and configuring a model of a system on a higher and non-programming level. A model author is usually not concerned and aware of low level and application specific aspects behind model construction such as the employed modelling methodology (see section 3.2). A model author usually concentrates on providing content, domain-specific semantics or pedagogical aspects.[4] An author can in most cases be expected to be a Subject Matter Expert (SME), who is most familiar with concepts and principles of the system domain.

Since SMEs are normally not computer experts and, therefore, cannot be expected to create and configure the simulation model on a low level, for example, encoding or modifying mathematical formulae on a program level, they need to be provided with tools and methods suitable for model authoring. Graphical user interfaces are most useful for SMEs by being intuitive to operate and by providing abstractions of the underlying low level knowledge engineering work. Some systems actually transparently produce program code to create the model.

Authoring tools for ICAI systems in general have to make a trade-off between power/flexibility versus usability (Murray, 1996). In connection with model-based simulation systems power/flexibility refers to the extent of the generality and openness of the tool and depth in terms of the depth and granularity to which the system can be modelled with the tool. Usability refers to the ease of learning and productivity of a tool when used by a SME.

3.8.1 Providing an authoring framework

Model creation in the context of the production of a complete training scenario can be split into two separate phases. The first phase is the basic design and implementation of the simulator by following a specific modelling methodology along with other design decisions. The simulator provides the basic structures and mechanisms in terms of a domain independent framework, also referred to as "shell", which then is kept unchanged. The second phase is the construction or authoring a domain specific model of the system on top of it. This can be done with considerably much less effort because the low level functionality are already provided by the underlying framework.

In practice, this distinction often cannot be made if implementation of the simulator framework and the specific model of a system is not clearly separated. This may be due to the fact that the whole process has been conducted in one step to speed up development, or parts of a model already exist, for example mathematical formulae are available and can be incorporated as in RBT (Woolf et al., 1986). Sometime, such practical aspects leave no space and time for a system design that potentially enables model authoring on top of it.

However, some of the model-based simulation training systems are targeted directly at model authoring and SME involvement in the training scenario production. A few other systems, usually

[4]Model authoring in a broader sense may also include authoring of instruction on top of the system model such as "recorded expert demonstration", authoring additional explanatory presentations or instructional planning. This is, for example, discussed in Coller et al. (1991).

aimed at prototypical and experimental use, have a potential and a general possibility of providing an authoring environment for it. These systems share some modularity and layering in the way they organise their domain knowledge according to the chosen modelling methodology.

Behavioural object-based systems such as IMTS or RAPIDS are designed to allow full SME participation in all model development stages, and are at the same time model authoring and model-based instruction delivery systems. The modelling methodology building on organisation and connectivity of objects can be captured in a graphical authoring environment. The SME assembles a model by selecting objects from an object library, adding them to the system model under construction, and finally, connecting them to other objects.

For simulator-based systems modelled with the Qualitative Approximation (QA) methodology such as Turbinia, no authoring environment has emerged up to now although it seems to be feasible to some extent. Due to the hierarchical structuring of the system and the modularity of the subsystems and components, the methodology seems to allow at least variations of steam cycles to be authored.[5] However, due to the QA methodology, low level quantisation and specific component connectivity, model reconfiguration on a low level in terms of parameter selection and adjustment has to be performed by model author. This has been reported to be not too difficult since the simulator itself is not very sensitive to parameter changes (Govindaraj, 1987). However, to the knowledge of the authors, the selection and adjustment of model parameters has not yet been formalised or automated in an integrated authoring tool.

3.8.2 Authoring on component level

An authoring environment for a system that constructs the domain model from low level components needs to provide authoring capability at this level of granularity. It must enable the model author to access and modify the properties of the component in an organised way. The mechanisms for basic communication between the components (such as a generic mechanism for transporting substance flows from one component to another) may be already provided by the modelling framework. The author is, therefore, required to provide only the semantics of the component, that is, its contribution to the domain model in terms of its behaviour, graphical representation and linkage to other components.

Object-based approaches can support authoring on component level. This is done by offering a library of graphical base-objects that can be used in the sense of either inserting an unmodified "instance" of the object into the simulation or starting from an already existing object, modifying it and then inserting the modified version into the simulation. The main requirement is that these library-objects are "context free" in that they do not make assumptions about their function in a broader context of the model. This ensures that the library objects can be reused in more than one special context.

In RAPIDS, the author creates a system model by selecting predefined parts from an object library and positioning them on the screen. Then, if required, rules are entered for the object to specify what values are passed between the objects in the model (Towne & Munro, 1992).

Appropriate tools for this bottom-up approach of constructing the whole model from small parts are required to support efficient creation of larger complex system models. These tools are needed to enable the author to assemble and verify, for example, complex objects from basic objects or to build up hierarchical structures within the model.

Authoring an object in isolation makes it sometimes difficult to predict how the object behaves in the system and in interaction with other objects (Towne & Munro, 1988). Therefore, the IMTS system editor contains a "breadboard" mode that makes it possible to connect several objects experimentally and observe their interactions. RAPIDS for example offers debug tools to determine the values of an objects attributes and its inputs and outputs (Merill et al., 1992).

[5]It should be noted that the Qualitative Approximation methodology in principle can be employed to model any dynamic system where the exact state values are not needed.

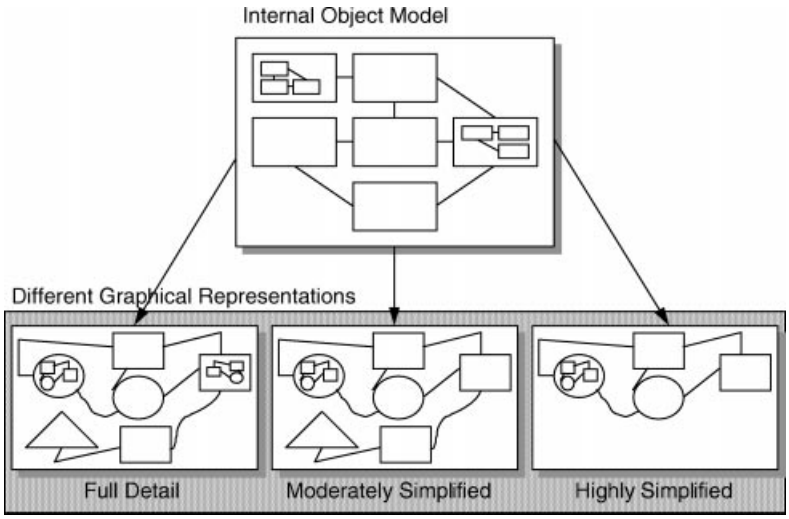

Figure 5 Different representations

3.8.3 Authoring different representations

From an instructional viewpoint, offering multiple representations with different levels of granularity are useful. Different representations are useful to adapt to the current instructional situation and to the current state of knowledge of the trainee. A simulation author has to manually create the different representations separately. In a system that is able to separate the graphical representation of a component from its role within the domain model, the additional model representations can be authored without much additional cost and effort. To construct different graphical representations of a domain model, the author must first create a complete and detailed domain model. The author can then remove or add graphical objects to the simulation to create the intended level of detail (see Figure 5). By leaving the underlying simulation unchanged and always executing the full model, accurate simulation is guaranteed.

With this approach representations can be simplified by, for example, leaving out entire subsystems or detailed functionality but at the same time still retaining the essence of the performed process. In IMTS Bladefold operation training scenario a hydraulic subsystem has been modelled in different degrees of simplification. A moderately simplified version of the subsystem shows the essential process, the retraction and extension of a bladefold cylinder plus peripheral functions that control the bladefolding process, while a highly simplified representation also leaves the peripheral functions out of the presentation and just shows the essential system operation (Towne, 1987).

The creation of simplified models requires the same domain expertise as the construction of the full detailed model. An author must be pedagogically aware as to which model simplifications are useful and correct regarding comprehension and consistency.

4 Discussion

4.1 General achievements of the field

The demand for ICAI simulation-based training is obvious. Simulations are needed to provide an adequate representation of a complex system on which training is to be conducted. In combination with appropriate instructional strategies both, individualised and comprehensive training can be achieved.

Various applications in this field have been built to address this need and have met many of the demands that arise in a training scenario of a complex system. To succeed, ICAI simulation-based training systems had to integrate methodologies and ideas from many different fields such as computer graphics, artificial intelligence, psychology or pedagogy, all research fields themselves.

ICAI model-based simulations for training have:

- demonstrated equivalence to traditional training settings under certain conditions,
- demonstrated the feasibility to provide high level authoring environments for non-computer expert SMEs to produce complex and sophisticated training simulations,
- demonstrated the feasibility of modelling a large dynamic system for diagnostic training through an appropriate modelling methodology,
- integrated a purely qualitative modelling approach with rather traditional methods to obtain a successful hybrid system,
- demonstrated the ability to couple an existing training system with a high-end VR visualisation technology,
- used graphical abstractions to support and foster mental models of the task domain.

4.2 General shortcomings

Simulation-based training ICAI projects have demonstrated their effectiveness with a variety of successful applications in various domains. The design approaches underlying these systems have been diverse. However, most of the developed applications often remained in a usable but prototypical stage, lacking thorough evaluation, proper verification or adequate application maintenance in terms of extensibility and reuse of existing software or adaptation to new technological platforms.

There is little consensus on system design of ICAI simulation-based systems for training. In the past, many systems have investigated and engaged in research on a limited number of research questions, and have produced prototypical results. The lines of development and research are often unrelated, and often do not build on each other. The result is that almost no frameworks or building blocks are available. Almost every research activity in the field has to start from scratch. This hinders sharing of results and gained expertise between research efforts. On the bottom line, these problems slow down the development of the whole field in a significant way.

In many cases these projects also fail to include SME domain expertise in the technical design of the system. This has a negative influence on the overall quality of the training system produced regarding consideration and integration of domain knowledge. Additionally, high development costs and other organisational issues lead to barriers in transferring training technology to industry (Bloom, 1996).

Steps suitable for evaluating ICAI systems and the need to move from internal laboratory to external field research and testing is described in Regian and Shute (1994).

4.3 Discussion of design approaches

4.3.1 Modelling methodology

Increasingly, research has shown that pure qualitative or quantitative modelling cannot produce an adequate simulation of a complex system for training. Pure Qualitative Modelling (e.g., Qualitative Physics) can play an important role when integrated as a specific and supportive part of a simulation-based training system (see CyclePad). It can be very useful to model general knowledge in terms of dependencies and constraints of processes in the real world. In CyclePad, for example, it has been used to detect impossible system designs by the trainee. Some researchers have observed that pure qualitative modelling does not scale up to real systems (Shankararaman & Lee, 1994a, 1994b; Shankararaman & MacFarlane, 1992). Pure Quantitative Modelling is suited to producing accurate simulation of a system but unable to clearly reason and explain the underlying model to the trainee, therefore is not very effective for training.

Object-based modelling is an open design approach since it still leaves room for further design decisions. It provides the model-author with a basic "technical" framework but imposes no semantics or hierarchies. The price for this flexibility is that it leaves much to be done for the

author. An author using this approach needs good tools for successful and efficient authoring of a complex system (see below).

QA imposes domain semantics in the form of hierarchies and system organisation. This limits flexibility but supports (relatively easy, because hierarchies are already given) construction of similar complex systems.

The choice of the modelling methodology has a significant influence on the other design features since it imposes a "base structure" of the model and with this defines and limits the amount, quality and accessibility of the information that is available from the model for the other components of the training system.

4.3.2 Simulator integration

Standalone simulation. Integration of the model-based simulator with the rest of the ICAI system is necessary. An unintegrated simulator can only have a very limited training capacity. They are unable to respond to the trainees current training situation and debug student errors and misconceptions. These efforts usually lead to discovery worlds where the trainee interacts with the simulator in a self-directed manner and with little or no coaching available.

However, as demonstrated by the STEAMER effort, stand-alone simulators are not limited only to microworlds, which just represent a discoverable and manipulatable domain, but are also open to impart knowledge about a domain for trainee exploration.

Object-based simulation. Approaches based on behaviourally described objects can be seen as a "distributed expert system"; a knowledge base of production rules distributed with the objects which form the simulator. A complete simulator of this kind would provide a stand-alone system available for discovery and free play. Objects could (as in STEAMER) be added that provide graphical explanations, and present knowledge about the domain in a limited form.

Additionally, as the object rules are flexible so that they could also be used to detect invalid or inappropriate settings (e.g., switches), an object-based stand-alone simulator could provide limited guidance and coaching by, for example, popping up a message window if the engine speed exceeds a certain limit.

An instructional component of a simulation-based ICAI system can access single objects (maybe via a naming scheme) of the simulation and query their states. This allows coaching and instruction to work directly on the object-states or combination of object states. Certain settings can be defined by combination of object states and used to indicate the initial start-setting or the end-setting of an instructional unit. The instruction is successful if the trainee is able to bring the simulated system to the defined end-setting. Additionally, sequences of operation on objects can be defined, possibly by expert demonstration.

The problem with the object-based approach is that it is very difficult to detect misconceptions, good moves and strategies by the trainee when considering only single object states. This problem can be partly overcome by considering higher level information on strategies or dependencies between object states (Towne, 1992).

4.3.3 Visualisation and interaction

Visualisation. Providing multiple views from different perspectives on the simulation model can be instructionally beneficial. The multiple perspectives have the potential to deepen the conceptual understanding of the complex system by allowing the trainee to develop and verify hypothesis about the concepts of the domain by allowing the trainee to view and compare information from various perspectives.

Additionally, graphical representations can be used to present knowledge about the domain. Visualisation of normally invisible processes or invisible substances can provide the trainee with an information-rich environment that is not usually available in reality. By enhancing visibility by, for example, visualising flow-rates within pipes, the trainee can better develop the understanding of the underlying domain concepts. Furthermore, explicit visualisation of model parameters is useful to

emphasise important variables of the domain in terms of their dependencies, how they are related to the overall system behaviour or their response to trainee interactions.

A high visual correspondence between real system and the graphical representation of the simulation model can reduce the cognitive load of the trainee and improve training. A high level of visual realism relieves the trainee from having to interpret the presentation and allows to concentrate on the training objectives. However, this need for realistic visualisation becomes increasingly unimportant as the trainee gets more and more familiar with the abstracted representation and concentrates more on the training task.

Training scenarios based on VR systems are capable of involving many of the human senses into the training process. Thus they are able to introduce a greater realism and the sense of self-location (''immersion'') into training. It has been stressed that these techniques provide the means for a more natural interaction in terms of cognitive interpretation and manipulation of the training environment, and thus result in a reduction of the cognitive load on part of the trainee. However, many open questions remain to be answered. How, for example, can authoring in such scenarios take place that exploits the full potential of an instructional environment; or how can trainee guidance be realised when the trainee is allowed to move around freely?

Interaction. The possibility to interact in terms of inspecting and manipulating the simulation model is a precondition and foundation for the active participation of the trainee in the instructional process. This active participation of the trainee in the instructional process contributes to the communication of knowledge and skill by allowing and encouraging the trainee to test and verify assumptions and hypotheses formed about the complex system. Additionally, active participation supports the paradigm of discovery learning, in which the trainee is, under certain constraints, free to explore the presented domain in a self-paced manner and incrementally discover facts and processes within the domain.

Model inspection and direct manipulation (associated with direct feedback) open the door for exploration and ''learning by doing''. Being able to practically test a hypothesis supports the development of conceptual (or mental) models (Wenger, 1987; Woolf et al., 1986).

4.3.4 Fidelity

Relate to training objectives. Research indicates that fidelity in the design of simulation-based training systems can be varied in order to be instructionally effective. To achieve suitable levels of fidelity for a given training objective, the operational setting as a whole needs to be considered and analysed (Hays & Singer, 1989). In case the interaction with a simulation model is the primary instructional interest, then high fidelity is very likely needed on the interactive hands-on part of the training system. In another case, where one of the training objectives is troubleshooting, possibly allowing replacement of suspected components, then a high structural fidelity is needed because a trainee needs to be able to relate causes and effects of faults to the structure underlying it.

Level of experience. Another important finding relates the level of experience of the trainee to the most effective level of fidelity (Alessi, 1988). It suggests that novice students tend to not benefit from very high fidelity since high fidelity means higher complexity which taxes memory and other cognitive abilities. Additionally, proven instructional techniques for initial learning tend to employ lower fidelity. As the student becomes more experienced, increasing the level of fidelity is useful.

4.3.5 Abnormal behaviour

Diagnostic training is one of the central elements of training with model-based simulations. Diagnostic training goes beyond the pure operation of a complex system by being able to simulate system failures or specific emergency situations in a realistic way comparable with the real system. It is, therefore, necessary to allow the modelling and the controlled exploitation of abnormal system behaviour.

The design approach to enable abnormal behaviour is closely related to the basic modelling methodology taken to create the complex system model. The underlying modelling methodology

enables or limits the depth and comprehensiveness with which the abnormal behaviour can be reproduced for training. A purely mathematical model, for example, is not able to provide insight into the abnormal behaviour at the level of system components because the structural information needed are not available. An equally important aspect is the ability to control fault effects in order to enable effective diagnostic instruction, especially if the fault can potentially introduce highly dynamic and complex abnormal behaviour.

Modelling level. As the specific modelling methodology imposes limits to the level and granularity of the modelling, it influences the modelling of abnormal behaviour as well. Therefore, modelling and presenting abnormal behaviour has to be considered on several levels. Looking at the abnormal behaviour of a complex system in terms of looking at a representation of the overall process (as in RBT) is a high level and rather abstract view. Looking at the dynamic interactions among abnormally behaving basic components of a complex system is a detailed low level view.

The required level of this view and approach for conducting diagnostic training depends on the training objectives. However, it should be considered what each view and approach-level to abnormal behaviour enables and what it does not enable. With a high level approach to modelling of abnormal behaviour it is not possible to switch to a lower and more detailed view on the abnormal behaviour (e.g., looking at interactions of subsystems or components). In contrast, it is possible to view abnormal behaviour modelled on a low level (e.g., on the component level) in a more abstract way. This is because the abnormal high-level overall behaviour evolves from the abnormal low-level interactions between subsystems or components.

So it should be noted that with the information available of abnormal behaviour modelled on a low level it is in principle (not negating the problems associated with it) possible to infer high level abnormal system behaviour while this is not possible vice versa as simply no low level information are available.

Modelling abnormal behaviour introduces additional complexity. Since the introduction of a fault or faults into a complex system model is usually related to an instructional objective (such as identifying the fault, identifying, selecting or actively executing corrective actions), the system must be aware of the effects of the fault and what actions exactly eliminate or generally have an influence on the fault. In systems where interactions between subsystems, components or parameter dependencies are complex and dynamic, the effects of faults and the effects of measures taken by the trainee are usually difficult to predict.

Therefore, to ensure a correct and instructionally reasonable system behaviour under the influence of one or multiple faults, usually additional elaborate measures specific to the basic modelling methodology have to be adopted. In many cases, this has led to building an additional knowledge base that stores information on effects of faults, dependencies of faults and their remedy. This process can be partially automated (by automatically inserting faults and recording their effects) as in the IMTS based Bladefolding training system or by manual configuration during system modelling.

4.3.6 Object orientation

Object orientation is a powerful software engineering approach that allows to map a real world view of a problem directly onto an implementation of the problem in software by focusing on the terminology, resources and abstractions that are present in the application world. Object-oriented design supports mapping the entities and types of the application domain onto the objects and classes of the solution domain (implementation design). It has been suggested that architectural aspects and in particular object-oriented approaches should be considered while developing training systems (Lesgold, 1994).

Object-oriented design can be the natural approach. In the design of simulation-based systems for training, object-based approaches can benefit from the object-oriented paradigm. These systems are built on components as separate entities which interact to produce the overall system behaviour.

These components can be mapped directly on to the objects in the solution domain according to the object-oriented paradigm. They can be arranged in an inheritance hierarchy to allow generalisation of functionality at a high level of abstraction and specialisation of object functionality at low levels. At an abstract object level, a component structure which is common to all objects could be defined. This definition could include communication protocols and data exchange between objects. Additionally, it can also include generic rules specific to the object. At a low level, the object can model a specific component. It could implement, for example, how the component has to draw itself on screen or specific conversions for its input and output data.

Additionally, it has been suggested that object-oriented programming can also be useful in modelling instructional aspects such as trainee coaching or assessing trainee performance and measuring knowledge transfer (see Lesgold, 1994).

Good object-oriented design is likely to result in higher value software. Training environments are often subject to change as the training domain slightly changes or the training objectives are altered. As training software has to match these requirements, the adaptability of the software in terms of modifiability and extensibility are important factors. In well designed object-oriented systems modifications can be in many situations kept local and restricted to a certain object class or an object layer in the system. If the system is extended to include, for example, a different graphical component of a binary switch, then the system can be easily extended by creating a new object that inherits all data and methods from the generic parent switch object, but implements its own specific method for drawing itself. Usually, object-oriented systems are extensible and allow reuse of existing code, thus help to avoid extensive re-implementation work and add to the maintainability of the software.

An essential criteria for the value of training software is the applicability across different domains. If the same software, or at least portions of software, can be used in different domains or domain settings, its value is potentially higher than software that has been produced and can only be applied to a very specific domain setting. Object-oriented design provides the means to design a system top-down, moving from a general to a problem-specific view on the domain. This enables and supports design approaches that are on their top level generic, e.g., offer only a framework with definitions of components, interlinkage of components and definitions for hierarchies, without being specific about a certain application domain. Such frameworks are, at least in part, applicable across a number of different domains and avoid full re-implementation of the training software.

Object-oriented programming promotes the reuse of existing designs and portions of code by concepts and mechanisms such as inheritance, polymorphism, abstraction and encapsulation. Recent studies have demonstrated that and how reuse in object-oriented software engineering improves both development efficiency and product quality.[6]

4.3.7 Authoring

Authoring support is crucial for acceptance. The availability of authoring tools that reduce complexity of the underlying knowledge base and motivate use and a reuse of system architectures is seen as a critical factor in enabling the training system to get wider acceptance in the industry (Bloom, 1996). This factor has not been fully addressed in most of the existing systems.

Authoring support consolidates the approach. We see the "formalising" effect on an authoring tool as very important. It has a consolidating effect for the taken approach by

- offering a sound and usable authoring support which is crucial for a wider acceptance and incremental development of a system (as suggested early on by Johnson, 1988);
- enabling SMEs, usually non-computer experts and unfamiliar with programming the system, to

[6]Evidence recently collected in an adjacent field (object-oriented system development) empirically proves the strong impact of reuse on product productivity and product quality (Basili et al., 1996).

contribute high quality and highly specialised expert knowledge (possibly in a ''rapid prototyping'' fashion) (Towne & Munro, 1991; Merill et al., 1992);
- forcing the development of the system to a stable plateau state which can then be used as a solid platform to further other research.

Complex simulation authoring requires good tools. The more open and unconstrained a model-based simulation system for training is regarding the domain and the structuring of content, the better the tools with which content can be created have to be. The tools must be flexible enough to support the SME in creation and verification of a variety of instructional material.

A training scenario modelled for example with the Qualitative Approximation methodology, would have to provide means for the SME to specify interrelated subsystems or system components in the system hierarchy and allow calibration of simulation parameters, etc.

Systems modelled from behaviourally defined objects neither impose a predefined hierarchical structure nor do they overtly constrain interactions and behaviour of the model components. As an object-oriented approach they allow an SME to create and work with single objects in isolation (free of a structural context). This enables the creation of libraries of context free objects. However, observing the behaviour of an object in isolation and predicting the behaviour and the effect of this object behaviour in interaction with other objects in a broader model context is often difficult. In such cases, additional tools for model development are needed to allow efficient calibration and verification of a simulation model during the creation phase, that is they must support model debugging.

5 Summary and outlook

We have presented a critical survey of the different design approaches underlying the construction of simulation-based ICAI systems for training. We have described the motivations and the basic technical background for using model-based simulation for training. For presenting an effective survey of these approaches, we have developed a framework which we think is suitable to separately address the important features of each design approach in an organised way. The design approaches have been described and illustrated with examples of existing model-based simulation applications in ICAI for training.

In the discussion, we have addressed the field in general. Based on the results of the survey, we have critically discussed several aspects of the design approaches, that are in our view of major importance.

Based on the results of the survey, we try to predict some of the forthcoming development trends in the field of model-based simulation ICAI systems for training. For the future we see the field of simulation-based ICAI systems generally developing in two major ways. Firstly, by exploring and integrating new technologies which are seen as potentially interesting for the training sector such as VR. Secondly, by employing advanced software engineering methodologies and more generic system architectures.

Simulation-based ICAI systems have always been successful in addressing enabling technologies and paradigms that have a potential for training by exploring their opportunities and integrating the technology in a prototypical way. Currently the opportunities of using VR in instructional simulation-based settings are explored. The potential of VR technology is great, however its opportunities, applicability and limitations for ICAI training are currently being investigated.

The construction of simulation-based ICAI systems are software-engineering efforts. Views on software engineering and development has dramatically changed over the last decade, taking a modular view on complex software systems. Software engineering is aiming at efficient production of stable applications by concentrating and supporting the role of objects, their interactions and layering. It is only appropriate that the new software engineering methodology and paradigms are considered when constructing a simulation-based ICAI system.

The ''front loaded'' nature of the development of simulation-based ICAI systems is one of the

main reasons for the current acceptance barrier in industry and blocks the widespread use of simulation-based ICAI systems. This problem could be solved with the creation of "shell" approaches in terms of reusable frameworks for system development.

The continuously growing infrastructures offering standards and technological platforms for global connectivity and allowing distributed and co-operative use of application scenarios may also influence the future of simulation-based ICAI training systems.

5.1 Virtual reality

It seems intuitively obvious that VR technology can be beneficially combined with model-based simulations for ICAI training. The effect of immersion in simulation-based ICAI is however not yet entirely clear. The few prototypical systems that have been designed up to now indicate the potential and the usefulness of the combination, but need further evaluation.

Evaluations still have to show to what extent the new opportunities available with VR can be beneficial for computer-based training. The question remains if or in which context VR simulation-based training can provide better instruction compared to traditional desktop-based 2D instructional applications. Up to now, it seems unclear if the metaphors and methodologies that have been developed in conventional 2D instructional systems and proven to be effective to, for example, support the development of mental models, can also be applied to VR-based instruction. However, it is clear that VR-based training can provide a new dimension of realism and presence which is certainly beneficial in many instructional contexts.

Additionally, since VR-based instructional scenarios are environments rather than simulators with an interface, they may support group training and synthetic agents as well as single-person training. This may move the focus away from the approach of a strictly individualised training towards some other paradigm, possibly with an emphasis on co-operation and communication between actors in a common virtual world.

5.2 Object orientation

Object-oriented techniques may have a positive effect for designing and implementing at least partially reusable computer-based training systems. It may have a normative effect because it separates design and implementation phases. This allows to potentially reusing system design and implementation separately.

On the practical side, most of today's application development work is done in an object-oriented manner. Environments and tools for every phase in object-oriented application development are available. They support object-oriented system analysis, design, implementation and maintenance. Software libraries are available that offer functionality to deal with various aspects of development such as GUI development or networking in a modular, reusable and often platform-independent way.

The strong emphasis of object orientation on system design and the available development support could lead to more generic and more robust system architectures, partial solutions, or to reusable layers and modules of software produced for simulation-based ICAI training.

5.3 Shells

To avoid the initial effort and high costs of building every new system from scratch, it is seen to be beneficial to establish an integrative framework providing basic functionality which can be reused. The framework can either be strict such as a "frozen" hierarchical model layering methodology (as in the Turbinia Vyasa system) or as open as an object-based methodology.

Such a "shell" can be a stable foundation for adding content on top of the framework. A "shell" approach generally has to find the right balance between implying too many restrictions to the content that can be created and the complexity and specialisation of tools needed to create contents.

On the one hand, imposing strict limitations on the type of domain or the involved hierarchies the "shell" approach may not be open enough. On the other hand, if the tools needed to create contents for the "shell" are too specific and complex, the applicability and usability of the framework across domains might be restricted and inefficient.

However, the use of "shells" has the potential to make the development of simulation-based training systems more efficient in terms of development time and system quality.

5.4 Distribution and co-operation

Network technology will influence model-based simulation ICAI systems for training. This may not necessarily imply major changes to the design approaches and methodologies (as discussed above) underlying the systems but force the systems to comply with standards (e.g., Java, WWW, Corba) that allow networked access, delivery and execution. Systems operating on these standards enable distributed, co-operative training and authoring scenarios. Frameworks for instruction with such characteristics are already available and in place (Boehm et al., 1996), and can be used to integrate and deliver the highly specialised model-based simulation ICAI systems.

References

Alessi, SM, 1998. "Fidelity in the Design of Instructional Simulations" *J Computer-Based Instruction* **15**.

Baron, S, 1987. "Application of the optimal control model to assessment of simulator effectiveness" in WB Rouse (ed) *Advances in Man–Machine System Research, Vol 3* JAI Press.

Basili, VR, Briand, LC, Melo, WL, 1996. "How reuse influences productivity in object-oriented systems" *Communications of the ACM* **39**(10).

Bennett, K, 1992. "The use of on-line guidance, representation aiding, and discovery learning to improve the effectiveness of simulation training" in J Regian, V Shute (eds) *Cognitive Approaches to Automated Instruction* Lawrence Erlbaum.

Bloom, C, 1992. "The application of Intelligent Tutoring Systems to Training Issues for People with Disabilities" Systems & AI Department, Honeywell Sensor & System Development Center.

Bloom, C, 1996a. "Roadblocks to Successful ITS Authoring in Industry" NYNEX Science & Technology Inc.

Bloom, C, 1996b. "Promoting the Transfer of Advanced Training Technologies" in C Frasson, G Gauthier, A Lesgold (eds) *Intelligent Tutoring Systems*, *ITS '96 Conference Proceedings* Montreal.

Boehm, K, Helfesrieder, B, Mengel, M, 1996. "Teaching and Learning using Java and the WWW" *Object World 96*, *Conference Proceedings*.

Booch, G, 1994. *Object-Oriented Analysis and Design with Applications* Benjamin/Cummings.

Brown, JS, Burton, R, de Kleer, J, 1982. "Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III" in D Sleeman, JS Brown (eds) *Intelligent Tutoring Systems*.

Coller, L, Pizzini, Q, Wogulis, J, Munro, A, Towne, D, 1991. "Direct manipulation authoring of instruction in a model-based graphical environment" in L Birnbaum (ed) *The International Conference on the Learning Science*, *Proceedings*.

Earnshaw, R, Gigante, M, Jones H (eds), 1993. *Virtual Reality Systems* Academic Press.

Fath, J, Mitchell, M, Govindaraj, T, 1990. "An ICAI Architecture for Troubleshooting in Complex Dynamic Systems" *IEEE Trans Systems, Man, and Cybernetics* **20**.

Fleming, J, Horwitz, C, 1996. "Applications of the Rapid Intelligent Tutoring System Development Shell" *Montreal ITS'96 Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs* Montreal.

Forbus, K, 1997. "Using Qualitative Physics to Create Articulate Educational Software" *IEEE Expert*.

Forbus, K, Whalley, P, 1994. "Using qualitative physics to build articulate software for thermodynamics education" *Proceedings of AAAI-94*.

Forbus, K, 1985. "Qualitative Process Theory" in D Bobrow (ed) *Qualitative Reasoning about Physical Systems* MIT Press.

Gigante, M, 1993. "Virtual Reality: Enabling Technologies" in R Earnshaw, M Gigante, H Jones (eds) *Virtual Reality Systems* Academic Press.

Govindaraj, T, 1987. "Qualitative approximation methodology for modelling and simulation of large dynamic systems: Application to a marine powerplant" *IEEE Trans Systems, Man, and Cybernetics* **17**.

Hays, R, Singer, M, 1989. *Simulation Fidelity in Training System Design: Bridging the Gap between Reality and Training* Springer-Verlag.

Helfesrieder, B, Shankararaman, V, 1998. "A framework for dynamic visualisation in simulation based instructional training" *System, Man, and Cybernetics Conference*: *IEEE SMC98*.

Hollan, J, Hutchins, E, McCandless, T, Rosenstein, M, Weitzman, L, 1987. "Graphic interfaces for simulation" in B Rouse (ed) *Advances in Man–Machine System Research*, *Vol 3* JAI Press.

Hollan, J, Hutchins, E, Weitzman, L, 1984. "STEAMER: An Interactive Inspectable Simulation-Based Training System" *AI Magazine*.

Hutchins, E, Hollan, J, Norman, DA, 1986. "Direct Manipulation Interfaces" in DA Norman, S Draper (eds) *User Centered System Design* Lawrence Erlbaum.

Johnson, J, Rickel, J, Stiles, R, Munro, A, 1998. "Integrating Pedagogical Agents into Virtual Environments" *Presence* **7**(6).

Johnson, J, Rickel, J, 1996. "Intelligent Tutoring in Virtual Environment Simulations" *ITS '96 Workshop on Simulation-Based Learning Technology*.

Johnson, W, 1983. "Pragmatic considerations in Research, Development, and Implementation of Intelligent Tutoring Systems" in M Polson, J Richardson (eds) *Foundations of Intelligent Tutoring Systems* Lawrence Erlbaum.

de Kleer, J, Brown, JS, 1985. "A qualitative physics based on confluences" in D Bobrow (ed) *Qualitative Reasoning about Physical Systems* MIT Press.

de Kleer, J, Brown, JS, 1983. "Assumptions and Ambiguities in Mechanistic Mental Models" in D Gentner, A Stevens (eds) *Mental Models* Lawrence Erlbaum.

Laurel, B, 1986. "Interface as mimesis" in DA Norman, S Draper (eds) *User Centered System Design* Lawrence Erlbaum.

Lesgold, A, 1994. "Assessment of intelligent training technology" in E Baker, H O'Neil (eds) *Technology Assessment in Education and Training* Lawrence Erlbaum.

Lesgold, A, Lajoie, S, Bunzo, M, Eggan, G, 1992. "SHERLOCK: A coached practice environment for an electronics troubleshooting job" in J Larkin, R Chabay (eds) *Computer-Assisted Instruction and Intelligent Tutoring Systems* Lawrence Erlbaum.

Loftin, R, Savely, R, 1991. *Advanced Training Systems for the Next Decade and Beyond* American Institute of Aeronautics and Astronautics.

Merill, M, Jones, M, Towne, D, Nason, E, 1992. *Functional Requirements of an Advanced Instructional Design Advisor: Simulation Authoring (Vol 3 of 3)* Brooks Air Force Base.

Miller, JR, 1988. "The Role of Human–Computer Interaction in Intelligent Tutoring Systems" in M Polson, J Richardson (eds) *Foundations of Intelligent Tutoring Systems* Lawrence Erlbaum.

Munro, A, Johnson, M, Pizzini, Q, Surmon, D, Wogulis, J, 1996. *A Tool for Building Simulation-Based Learning Environments* Behavioral Technology Laboratories, University of Southern California.

Munro, A, Pizzini, Q, 1996. *RIDES Reference Manual* Behavioral Technology Laboratories, University of Southern California.

Munro, A, 1993. "Authoring Interactive Graphical Models for Instruction" in D Towne, T de Jong, H Spada (eds) *Simulation-Based Experiental Learning* NATO ASI Series F, Vol. 122, Springer-Verlag.

Murray, T, 1996. "Having it all, maybe: Design tradeoffs in ITS authoring tools" in C Frasson, G Gauthier, A Lesgold (eds) *Intelligent Tutoring Systems*, *ITS '96 Conference Proceedings* Montreal.

Psotka, J, 1985. "Immersive Tutoring Systems: Virtual Reality and Education and Training" U.S. Army Research Institute.

Rasmussen, J, 1985. "The role of hierarchical knowledge representation in decisionmaking and system management" *IEEE Trans Systems, Man, and Cybernetics* **15**.

Regian, J, Shute, V, 1994. "Evaluating Intelligent Tutoring Systems" in E Baker, H O'Neil (eds) *Technology Assessment in Education and Training* Lawrence Erlbaum.

Rickel, J, 1989. "Intelligent computer-aided instruction: A survey organized around system components" *IEEE Trans Systems, Man, and Cybernetics* **19**.

Rothenberg, J, 1989. "The nature of modeling" in L Widman, K Loparo, N Nielsen (eds) *Artificial Intelligence, Simulation and Modelling*.

Schut, C, Bredeweg, B, 1996. "An overview of approaches to qualitative model construction" *The Knowledge Engineering Review* **11**(1).

Shankararaman, V, Lee, BS, 1994a. "Knowledge-based safety training system: A prototype implementation" *Computers in Industry* **25**.

Shankararaman, V, Lee, BS, 1994b. "Integrated approach to offshore safety training" *Trans Institute of Marine Engineers* **106**(3).

Shankararaman, V, MacFarlane, CJ, 1992. "Problems with training simulators" *Proceedings of the Institute of Marine Engineers and Royal Institution of Naval Architects, Joint Offshore Group International Conference on Offshore Safety*.

Stevens, A, Roberts, B, 1983. "Quantitative and Qualitative Simulation in Computer Based Training" *Computer-Based Instruction* **10**.

Towne, D, 1995. *Learning and Instruction in Simulation Environments* Eductional Technology Publications.

Towne, D, 1993. "Teaching and learning diagnostics skills in a simulation environment" in D Towne, T de Jong, H Spada (eds) *Simulation-Based Experiental Learning* NATO ASI Series F, Vol. 122, Springer-Verlag.

Towne, D, Munro, A, 1992. "Supporting diverse instructional strategies in a simulation-oriented training environment" in J Regian, V Shute (eds) *Cognitive Approaches to Automated Instruction* Lawrence Erlbaum.

Towne, D, Munro, A, 1991. "Simulation based instruction of technical skills" *Human Factors Society* **33**.

Towne, D, Munro, A, 1988. "The intelligent maintenance training system" in J Psotka, LD Massey, S Mutter (eds) *Intelligent Tutoring Systems – Lessons learned* Lawrence Erlbaum.

Towne, D, 1987. "The generalized maintenance trainer: Evolution and revolution" in WB Rouse (ed) *Advances in Man–Machine System Research*, *Vol 3* JAI Press.

Vasandani, V, Govindaraj, T, 1995. "Knowledge organisation in intelligent tutoring systems for diagnostic problem solving in complex dynamic domains" *IEEE Trans Systems, Man, and Cybernetics* **25**.

Wenger, E, 1987. *Artificial Intelligence and Tutoring Systems* Morgan Kaufmann.

Widman, L, Loparo, K, 1989. "Artificial intelligence, simulation and modeling: A critical survey" in L Widman, K Loparo, N Nielsen (eds) *Artificial Intelligence, Simulation and Modelling*.

Williams, M, Hollan, J, Stevens, A, 1983. "Human reasoning about a simple physical system" in D Gentner, A Stevens (eds) *Mental Models* Lawrence Erlbaum.

Woolf, B, Blegen, D, Jansen, J, Verloop, A, 1986. "Teaching a complex industrial process" *Coins Technical Report 86-24*, Computer and Information Science, University of Massachusetts, Amherst.