Using Hypertext in Developing the Human Computer Interface to Gaming-Simulation Environments that Incorporate Intelligent Tutoring Support

Marios C. Angelides and Ka Yan Tong

Information Systems Department, London School of Economics and Political Science, U.K.

A Gaming-Simulation Environment for teaching and learning that incorporates Intelligent Tutoring Support also serves as a supporting tool for the Human Computer Interface. The Human Computer Interface to this environment will affect how players interact with the domain that is both being tutored upon and is being the object of game play. The development of this interface is subject to a variety of constraints. This paper suggests a Hypertext Approach to developing this Integrated Environment and thus the Interface that promises to oust or minimise these constraints.

Keywords: Human Computer Interface, Gaming-Simulation, Intelligent Tutoring Systems.

1. Introduction

The adaptation of an Intelligent Tutoring System (ITS) in a man-machine Gaming-Simulation Environment to enhance the pedagogical effectiveness of such an environment as a teaching and learning tool will provide such an Integrated Environment with Intelligent Tutoring Support, which is a decisive factor for the successful implementation and operation of the environment [Angelides, 1994]. The HCI to this environment should be responsible for specifying or supporting not only the activities that the player carries out, but also the methods available to the player to perform those activities. Therefore the HCI defines the kind of problems the player is to solve as well as the tools available for solving them. The HCI in many ways defines the way the player looks at the subject matter.

The HCI to this environment affects two aspects of such an integration. First, it determines how players interact with the Environment. Second, it determines how players interact with the domain that is both being tutored upon and is being the object of game play. This is done either through the simulation of the domain during game play or the indirect connection to the domain itself during tutoring. This interaction is tied closely to the tutorial component of the system so that actions during game play are analyzed and acted upon.

A hypertext-based HCI may be used to address both of the aspects mentioned above. These aspects are seen to be crucial to the successful operation of such an Environment [Bielawski and Lewand, 1991]. With a Hypertext-based HCI the player can have a feeling of working directly with the domain: such HCIs are designed so that the actions and objects relevant to the task and domain map directly to actions and objects in the HCI. A Hypertext-based HCI allows a player to carry out desired computations by manipulating objects. The underlying mechanism behind such HCIs are almost always icons. If the inherent functionality is not self-evident to the inexperienced player, Hypertext-based HCIs allow the player to interact with the domain by giving commands to a computerised

intermediary, which then carries out the desired actions.

This paper proposes a hypertext approach to the development of the HCI to a specific Business Gaming-Simulation Environment which incorporates Intelligent Tutoring. The short-term aim of this paper is to show how the HCI issues that arise from such an integration can be addressed effectively using hypertext. The long-term aim is to highlight the promises of a hypertext based HCI to such an environment. The paper first gives an overview of intelligent tutoring systems (ITSs). It then discusses the HCI to the Integrated Environment, in particular the constraints that will affect the human-computer interaction, followed by a presentation of the 'metal box' business game. The paper then proceeds to suggest a hypertext approach to, and proposes the architecture of, an Integrated Environment for the 'metal box' business game. Future prospects for a hypertext based HCI are discussed in the concluding section.

2. Intelligent Tutoring Systems (ITSs)

For a Tutoring System to be classified as Intelligent, it must pass three tests of intelligence [Angelides, 1992]. First, the system must know the subject matter well enough to be able to draw inferences or solve problems in the domain of application. Second, it must be able to deduce a user-learner's approximation of the domain knowledge. Third, the tutorial strategy must allow the system to implement strategies that reduce the difference between the expert and the student performance. Therefore, at the foundation of an ITS one expects to find three special kinds of knowledge: domain, student, and tutoring knowledge.

The first key place for intelligence in an ITS is in the knowledge that the system has of its subject domain [Anderson, 1988]. There are three approaches to encoding knowledge into the domain model which gives rise to the three different types of domain models. The first approach, which gives rise to a black box model of the domain knowledge, involves finding a method of reasoning about the domain that does not actually require codification of the knowledge. A black box model generates the correct input-output behaviour over a range of tasks and so can be used as a judge of correctness. However, the internal computations by which it provides this behaviour are either not available or are of no use in delivering instruction. Such a domain model can be used in a reactive tutor that tells the students whether they are right or wrong and possibly what the right move would be. This is known as surface-level tutoring. The second approach, which gives rise to a glass box model of the domain knowledge, involves reasoning about the domain by applying codified knowledge. A glass box model is the standard knowledge based systems approach to reasoning with knowledge. Because of its nature, the emerging system should be more amenable to tutoring than a black box model because a major component of this expert system is an articulate representation of the domain knowledge. The third approach, which gives rise to a cognitive model of the domain knowledge, involves making the domain model a computer simulation of human problem solving in the domain of application.

Both generations of Expert Systems are currently in use in the development of Domain Models for ITSs. First generation Expert Systems can be developed following the criterion-based approach: any system that achieves high quality performance could be classifed as an Expert System. Therefore, any system capable of undertaking a complex task proficiently is regarded as an Expert System by the criterion-based approach. Such are the black box models of domain knowledge. However, first generation Expert Systems are mainly developed using the Knowledge Engineering methodology which involves deploying humanlike knowledge that is codified using one or more of knowledge representation schemes, mainly production rules, usually stored separately in a knowledge base. Expert performance is achieved through reasoning with the contents of such a knowledge base. Such are the glass box and cognitive models of domain knowledge.

Second generation Expert Systems which promise a more fundamental understanding of their domain of discourse and are not so narrow or brittle as their predecessors, are currently under test and development. They have not yet achieved the same levels of performance as the first generation of Expert Systems but they are regarded as the hope for the future. Expert Systems of this kind are developed as Qualitative

Process models which are concerned with the knowledge that underlies our ability to mentally simulate and reason about dynamic processes. This involves reasoning through the causal structure of a system. This is inferred by examination of the local interactions between system components and not of their function. As a result, the principle on which they work is called 'no function in structure' principle. Such are the cognitive models of domain knowledge.

A classic case of a glass box model of domain knowledge is that of GUIDON [Clancey, 1987] which uses the MYCIN expert system [Shortliffe, 1976]. MYCIN consists of 450 if-then rules which encode the probabilistic reasoning that underlies medical diagnosis. Tutorial interaction is driven by t-rules which are an extension of issue-based tutoring. However, t-rules refer to the internal structure of the domain model, such as rules and goals, and not on surface behaviour. T-rules are compiled to be the combination of the difference between what would be the expert behaviour and the student behaviour and expert reasoning process.

A classic case of a cognitive model of domain knowledge is that of WHY [Stevens, Collins and Goldin, 1982] which uses a schema representation for evaporating knowledge. There are slots for the actors of evaporation, for the factors that influence the amount of evaporation, for the function relationships among these factors, and the result of evaporation. Bugs are created by erroneous entries in the slots. WHY uses a set of tutoring rules for implementing the Socratic method of discourse. These rules have a resemblance to the issue-based recognition rules that are normally used for black and glass box models. However, here the conditions for such rules refer to the underlying knowledge rather than to any surface behaviour and incorporate a mixture of knowledge assessment and instruction.

The second key place for intelligence in an ITS is in the knowledge that the system infers of its student [VanLehn, 1988]. An ITS diagnoses a student's current knowledge of the subject matter and uses this to individualise instruction according to the student's needs. The ITS component that holds the student's current knowledge is the student model. The input for diagnosis is garnered through the interaction with the student. The output of diagnosis depends on the use of the student model. Nevertheless, it should reflect the student's current knowledge state. Common uses for the student model include advancing the user to the next curriculum topic, offering unsolicited advice when the student needs it, generating new problems, and adapting explanations by using concepts that the student understands. A student model usually consists of three kinds of information: bandwidth (i.e. quality and amount of student input), the type of domain knowledge (i.e. declarative, procedural or causal) and differences between the student and domain models in terms of missing conceptions (i.e. as an overlay model) and misconceptions (i.e. as a list of bugs).

The third key place for intelligence in an ITS is in the principles by which it tutors students and in the methods by which its applies these principles [Halff, 1988]. Tutor models may incorporate many different instructional techniques. A tutor model must exhibit three characteristics: (a) It must exercise some control over curriculum, that is, the selection and sequencing of material to be presented to the student, and some control over instruction, that is the process of the actual presentation of that material to the student, (b) it must be able to respond to student's questions about the subject matter, and (c) it must be able to determine when students need help in the course of practising a skill and what sort of help is needed. Some tutors are primarily concerned with teaching factual knowledge and inferential skills. These are the expository tutors. Some tutors are primarily concerned with teaching skills and procedures that manipulate factual knowledge. These are the procedural tutors. Curricula can be broken down into, formulating a representation of the material in the domain model and selecting and sequencing concepts from that representation. A tutor model must also incorporate some form of propaedeutics, that is knowledge which is needed for enabling learning but not for achieving proficient performance. The underlying assumption is that skilled performance will be achieved only with practice. As a result, propaedeutics serve, firstly, to relate theory to practice, secondly, to justify, explain, and test possible problem solutions, thirdly, as a stepping-stone to more efficient problem-solving strategies and, fourthly, as strategies for management of the working memory during intermediate stages of learning. Curricula serve several functions: (a) they divide the material to be learned into manageable

units which should address at most a small number of instructional goals and should present material that will allow students to master them, (b) they sequence the material in a way that conveys its structure to students, (c) they ensure that the instructional goals presented in each unit are achievable, and (d) they enable the tutor model to evaluate the student reaction to instruction on a moment-to-moment basis and for reformulating the curriculum.

3. The HCI to the Integrated Environment

Two assumptions are made about the users (would-be players) involved. The first one is that they come to the HCI with knowledge that will guide their use of that interface [Burton, 1988]: knowledge about their past use of the environment, about the kinds of real-world objects that might be manipulated by the environment, and about the kinds of real-world objects that might be portrayed and manipulated as part of the interface. This knowledge plays an important role in human-computer interaction. People combine this knowledge along with their observation of the structure and behaviour of the interface to construct a conceptual model of the environment. This model can then guide the user's interaction with the environment. A user will make reasonable guesses, about likely ways to handle novel problems, about probable reasons for errors, and about good ways to recover from errors.

The second assumption is that users come to the environment with a set of task-level goals that they want to achieve. Nevertheless, there is often a considerable distance between this goal statement and the actions that most interfaces make available to users. Spanning this gap (i.e. solving the external-internal task-mapping problem) poses a major problem for interface design and a major interaction problem for users. The greater the gap, the more difficult the interface will be to use. This gap can be minimised by designing the interface so that the actions supported by the interface map directly to corresponding actions in the domain. In principle, if users understand the domain, the use of the interface will be trivial. While the task-mapping problem cannot be ignored, the increasing availability of graphical user interfaces for interface design may help to reduce the effort needed to implement these specialised interfaces.

The interaction between students and the Integrated Environment is inherently complex because the users of this environment are, by definition, working with concepts they do not understand well. Consequently, a well-designed interface can add considerably to the way in which the student will conceptualise the game as such and as a problem domain, as well as over the vocabulary the student will use to talk about it. Human interface techniques will address two aspects of such an Environment. First, they determine how students interact with the Environment. A well-designed human interface allows the Environment to present instruction and feedback to students in a clear and direct way. Similarly, it can provide students with a set of expressive techniques for stating problems and hypotheses to the Environment. Second, they determine how students interact with the game or the domain that is being tutored, through either the simulation or direct connection to the domain itself. Both interactions are tied closely to the tutorial component of the Environment so that actions in the domain are analyzed and acted upon.

The quality of the HCI, i.e. its power, ease of use, and ease of learning, does not lie with its outward appearance but with the underlying structure of the interface and the Intelligent Tutoring Support behind it. What is important is not how the interface looks but how allows the user to understand the capabilities of the Environment. Since these capabilities are inherently constrained by the game and domain of discourse, it is important to have an interface that conveys the important properties and semantic constraints of the domain of the game. Therefore, the interface to this environment has to enable users to both become direct participants in the domain (first-person mode) and to control the domain by instructing an intermediary to carry out actions in the domain (second-person mode).

In first-person interface mode or direct manipulation interface mode, the user has a feeling of working directly with the game domain [Miller, 1988]. This mode allows a user to carry out desired actions by manipulating objects. This

interface mode is designed in such a way so that the actions and objects relevant to the task and game domain map directly to actions and objects in the interface. The underlying mechanism behind such an interface mode are almost always icons. Icons are small pictures on the screen which trigger some action when selected by the user. Icons represent data structures and procedures, and links between these objects specify how the procedures are to be applied on the data. Although first-person interface mode appears to offer significant advantages to users, some aspects of the Environment's functionality may not be self-evident to the inexperienced user. In such cases, the Environment will have to explain the different capabilities of the Environment to the user through its ITS support. Furthermore, the link between the semantics of the game domain and the semantics of the interface may be fuzzy. The problem here is how much of the underlying application is conveyed through this interface mode for the users to understand and which parts of the Environment they can directly manipulate.

With second-person interface mode the user interacts with the domain of the game by giving commands to a computerised intermediary, which then carries out the desired actions [Miller, 1988]. *Command Languages*, typical second-person interface mode, are keyword-oriented interface modes in which a command consists of a string of words and sometimes special characters that, when processed by the Environment's command interpreter, specify the action the user wants to carry out. With *Menus*, a list of options is shown to the user, who then selects the desired option by striking a key. Menu-based interface modes stand between first-person and second-person interface modes: being presented with information and selecting some of this information is a characteristic of second-person interface mode whereas the direct way in which the user can specify the information is characteristic of first-person interface mode. With a *natural language interface mode*, the most popular user interface mode to such an Environment, users communicate in a language they already know with an agent that can interpret their requests for action to be triggered. Human computer interaction in natural language is normally restricted to some form of stylised English. Full coverage is difficult because natural language interface modes are second-person interface modes in which the style of interaction is that of speaking to an intermediary who will carry out the requested actions.

3.1. HCI Constraints

Some of the constraints that may affect the human computer interaction in the interface to the Integrated Environment are discussed below [Angelides, 1992].

Task constraints. This pertains to breaking up a semantically rich game into meaningful components that are related to each other and embodying them in the HCI in order to reflect the semantic constraints of the domain of the game and at the same time help users control, understand and manipulate them.

User constraints. and *cognitive limitations* This involves compensating in the HCI for weaknesses in the users' cognitive abilities. Rather than requiring users to infer or guess the effects of their actions, these effects can be made an explicit and visible part of the interface. This is particularly useful where the user does not have a sound understanding of the information being worked with.

Instructional constraints. The domain of the game being tutored and the instructional role played by the Environment through the integrated ITS drive the form of the HCI. For example, instruction may entail continuous dialogue between the student and the environment, coaching that monitors user actions and offers advice about more efficient use of the environment would not, etc. The interface will be capable of conveying the information that is relevant to the instructional task and the pedagogy that takes place. Consequently, a second instructional issue is how the HCI presents this information to the student-user in a way that emphasises their most important properties.

Physical constraints. For a tutorial task that deals with a real world device or environment a highly realistic physical depiction along with a separate cognitive description of the behaviour of the device or environment and equally realistic techniques for interacting with this interface may be required for the student to understand and learn the domain. Students must understand the concepts involved but these should be presented in relatively abstract ways.

Tutorial constraints. This involves the extent to which the interface can ease the diagnosis and remedial tasks of the integrated ITS. First-person interface modes will present students with a realistic picture of the application modelled that provides a meaningful context for explaining and discussing the domain of the game and a student's possible problems with it. First-person interface modes rely on the manipulation of semantically rich objects and thus giving rise to student high level actions that are semantically rich. The interface may also include buggy objects to help remedy misconceptions. The integrated ITS can influence the capabilities of the interface in order to ensure that the user is carrying out semantically acceptable actions. Consequently, major classes of user errors could be eliminated by preventing the user from making them. Alternatively, the integrated ITS can identify misconceptions and help a student correct them.

Implementation constraints. The Integrated Environment is three things: an educational instrument, an interface to an application, and a simulation game. The issues related to the latter two aspects focus on how this environment may be implemented. Efforts into interface implementation takes a twofold approach. First, we must separate the interface of the Environment from the more direct computational part thus making the interface responsible for translating a user's interface actions into whatever form is required by the Environment. Second, we must expand the communication path between the interface and the environment and allow some communication between the Environment and its interface.

Knowledge: What is being learned? Another dimension along which HCI differs is the question of what is being taught and consequently what the student is learning. The knowledge a person has about a domain consists of facts about the domain, skills or knowledge of procedures in the domain, concepts that organise the facts and procedures in the domain. In addition, the person has meta-skills that aid in the learning of new skills. Different HCIs focus on supporting the teaching of different aspects of the knowledge a person should have about the domain, by changing the activities and tools in the environment.

Level of abstraction. The level of abstraction at which the knowledge is presented i.e. what features of the real world to represent and why is crucial.

Fidelity. Another dimension is the Fidelity of the HCI, i.e. the concept of how closely the simulated environment in the HCI matches the real world. A high fidelity simulation is one that is nearly indistinguishable from the real thing. Researchers have identified four different kinds of fidelity: physical fidelity (feels the same), display fidelity (looks the same), mechanistic fidelity (behaves in the same way), conceptual fidelity (is thought of as the same) and expert fidelity (how the methods used by the student and the domain expert to solve a problem correspond). By studying the difference between expert and novices in different domains, cognitive psychologists have discovered that students go through different conceptual stages in learning a subject. Related work has shown that students have the ability to learn advanced theories well enough to pass tests but still act in the real world in ways that contradict theories. As a result, learning is not pouring knowledge in an empty vessel but a process of reconceptualisation, of getting students to construct the appropriate knowledge out of the knowledge that they already have. It is important to take students through a progression and not merely to teach them the expert's notion, because it is through this progression that new knowledge connects to student's experiences of the real world. Combining the idea of semantic distance, i.e. the distance between the concepts that the HCI uses and the ones that the user has, with the movement of conceptual structures that takes place in the development from novice to expert leads to a conclusion that the HCI that is cognitively accurate for the novice will not be for the expert and vice versa. The conception of the domain in the HCI may need to change to create intermediate stages to move the student to a final state.

Sequences. This dimension views the student as being exposed to a sequence of increasingly complex micro-worlds that provide intermediate experiences such that within each micro-world the student can see a challenging but attainable goal. An important aspect of the instruction is the instructor's choice of the micro-world.

Help provided. Another dimension is the Help provided by the HCI to further the instructional process. There are different ways in which the environment may provide help to the student. The first is traditional help: the environment has help available upon request or during errors. The second is assistance: the environment does part of the task, sometimes the whole task. The third is empowering tools: with them the environment performs bookkeeping tasks that aid learning. The fourth is reactive: the environment reacts to student's ideas. The fifth is modelling: the environment performs that task while the student watches. The sixth is coaching: the environment breaks in and makes suggestions and the seventh is pure tutoring: the environment maintains control over interaction.

Structure provided. The final dimension of HCI is the amount of structure they impose on the activities they engage the student in. In one case it may be very unstructured. The HCI is carefully designed to embody a set of ideas and concepts, and students are allowed to explore it. Unstructured HCI are based on the belief that by providing a rich HCI worthwhile learning will emerge if students are encouraged to explore whatever interests them. In this case the HCI could help the student see and appreciate the ideas and concepts the integrated environment had to offer. Unfortunately, the Environment's ability to recognise interesting situations occurring in the student's activity is limited by how well it can understand what the student is doing. A structured HCI may provide an incomplete characterisation of the knowledge learned from it. Then the environment may work against the students' learning those issues that have been left out, and the students may fail to learn everything they need. Having a domain expert that performs a task does not guarantee that all of the requisite knowledge has been identified.

4. The Metal Box Business Simulation Game

The Metal Box Business Simulation Game [CRAC, 1978] was developed to give students an insight into the work of business managers and thereby acquire an understanding of business management. The Metal Box Company is a manufacturer and supplier of central heating boilers and radiators. The company has identified a widespread demand within the EC for a large, well-made domestic boiler and has designed the 90/120 'BTU Vulcan Continental' de luxe. These are sold to wholesalers in batches of eight boilers per batch.

The player (or group of players) starts business as one of the managers of the Metal Box Company having to solve financial, production or marketing problems. The business must be run efficiently to be able to pay for salaries, materials and services and to cover the costs of the development of new production resources, such as an expansion of the size of the factory. Additionally, the business has to yield a surplus to make a reasonable profit. The company has up to four players who appoint themselves to one of four roles. The *Production Director* has to make decisions on the amount of boilers to produce and has to determine the selling price. The *Sales Director* makes decisions on any market research or research and development to be undertaken, the number of sales persons to be recruited and which customers the sales staff should call on. The *Financial Director* has to master all calculations and is responsible for completing the Company's accounts. The *Managing Director* is in overall charge and he is the final arbiter on all operating decisions.

There are other companies which are proposing to make and sell similar equipment and therefore are competing in the market. Each company is assumed to have a starting capital to spend on producing and marketing the equipment. Further capital is available on loan from the bank. To enable the players to plan, make decisions and control operations, each company uses two documents: the Company Decision Sheet (CDS) and the Company Operating Statements (COS).

The Company Decision Sheet (CDS) records quarter by quarter all decisions made and indicates the situation at any time. The Production Director notes the decisions made as follows: simple decisions (e.g. size of the factory to be built, selling price), state of operations resulting from his decisions (e.g. notional reduction of the selling price), progressive operations (i.e. the process of the construction of a new factory, materials from ordering through work-in-progress to selling). Additionally, the CDS records the decisions made by the Sales

Director, such as the recording of progress of sales staff from recruitment through training to selling. The CDS therefore instantly gives information about when a factory is ready for production, when a product is ready for sale or when a salesperson is ready to go out and sell.

The Company Operating Statements (COS) contains the three main control accounting statements. They represent the state of a company's finances as follows: the Revenue and Expenditure Statement gives continuous monitoring of profitability quarter by quarter; the Cash Statement shows whether a company has enough cash to carry on in the next quarter and whether it needs to borrow from the bank; and a Summary Statement of Assets and Liabilities (according to the Balance sheet). The Financial Director is responsible for compiling this information, working out the position at the end of each quarter's operations. Most entries into the COS are derived from the CDS.

5. The Hypertext Development Approach

The proposed integrated environment will be specified using Hypertext, with some terminology borrowed from HYPERCARD II. Hypertext can be described as the creation and representation of links between pieces of data which can be text, graphics, audio, video, and or animation [Nielsen, 1990a]. Conceptually, the notion of hypertext is closely related to that of semantic networks, with data being represented in nodes. Thus hypertext is the organisation of information into information nodes and links [Halasz, 1988]. Information nodes are linked via information links either sequentially, hierarchically or mixed. The development procedure often follows an underlying object-oriented formalism. The framework usually designates the way in which information is to be held in a hypertext environment [Nielsen, 1990b] [Smeaton, 1991].

Generally, the main component of a hypertext environment is the stack [Angelides and Gibson, 1993] [Angelides and Tong, 1994]. Stacks can exist independently or be interwoven using links to form an environment. They are composed of a number of cards (by analogy to the more familiar index cards) which represent the information nodes. The stack is basically a collection of related nodes of information united because they are seen to logically belong together. Hypertext environments are usually developed in three distinct stages. Firstly, the information to be utilised in the environment has to be decomposed and organised into semantically meaningful cards and appropriate stacks. Secondly the links between the cards, cards and stacks, and between stacks must be installed. Thirdly, the stacks are linked together to form the hypertext environment. The link is the core of all hypertext environments. The link provides a path from one part of the hypertext environment to another.

With a hypertext framework for specifying the Environment, the domain, tutoring and student knowledge must first be organised into stacks of cards [Angelides and Gibson, 1993] [Angelides and Tong, 1994]. Secondly, links will be installed to integrate the stored knowledge, not only within individual, but also between models. For example, the contents of the student model should point to the "best" for the student teaching strategy in the tutoring knowledge model and to those goals that have been attained by the student and those that are yet to be attained. The student model should also point to those domain knowledge parts that have been mastered by the student, perhaps including a measure of the level of mastery, for example through an overlay model. The student model should also point to those domain knowledge parts that are the source of misconceptions for the student. In the contents of the model of the tutoring knowledge, goals which the environment will try to attain should point to that part of the domain knowledge that contains domain knowledge relevant to the goal, along with a pool of appropriate teaching strategies that would enable this. Once cards and stacks have been linked together the environment is ready for use.

All the knowledge of the environment will be incorporated in semi-structured hypertext information nodes (i.e. cards) which are in turn incorporated in stacks. The semi-structured kind of node is chosen because of its ability to allow labelled fields (i.e. attribute slots) and their values to be stored inside the node [Angelides, 1992]. A semi-structured information node has attribute slots which either have default values, or may be instantiated with specific occurrence

values. They may also have procedural attachments which are executed whenever a value is needed or changed.

A card may be linked to other cards with which it is related by a class-instance relationship. This establishes a semantic network of cards which are organised hierarchically so that properties can be inherited from generic cards (i.e. cards higher in the hierarchy) to cards lower in the hierarchy. Thus, a card which represents a concept can be decomposed in its hierarchical constituents, and allow these to inherit all its properties. Related cards may be stored in a stack. There will be links from the slot values to other cards with which a given card is hierarchically related in this hierarchical network. These links will be set up as "organisational" hypertext information links to connect a parent card with its children and thus establish a hierarchical tree in this hypertext network.

Cards are also linked to other cards with which they are not hierarchically related via "referential" hypertext information links and thus establish a non-hierarchical structure in this network [Angelides and Gibson, 1993] [Angelides and Tong, 1994]. Any information related to a card which cannot be included in the card structure, will be "annotated" to the card as a "typed" hypertext information node, if it is text, or as a "graphical" hypertext information node, if it is an image, or via an "annotation" link. This will establish a part-to-whole relationship with a given card. Within this annotation, there may be further referential, keyword or annotation links to cards which the annotation may relate to. Finally, a card may be linked to another card by a "keyword" link, if two cards have the same value for a given attribute slot. The link names will carry a name which will depict semantic information.

6. Developing a Gaming-Simulation Environment for the "Metal Box" Game that Incorporates Intelligent Tutoring Support

We will proceed with our proposal for an architecture by embedding an ITS within a gaming-simulating environment [Angelides and Paul, 1993a]. In this case, the ITS will completely take over the role of the game tutor, i.e. it will take over all four standard operations that will have to be performed in the game, and it will include all the knowledge that is necessary to enable the ITS both to run the game and to provide tutoring about the game. Consequently, the ITS will be responsible to see that the rules and roles are understood and observed, to draw attention to, and explain, any difficult points, to progress the playing of the game, to compare decisions by competing companies, to draw out lessons about business planning and control, to check the calculations on the operating statements, to allocate orders according to the rules, to provide market research information, if requested, to fix the market size and to allocate customers' order potential.

The main components in the Environment's architecture are a domain model, a tutor model and a student model [Angelides and Paul, 1993b]. The domain knowledge model includes all the knowledge about the game and, therefore, will be used both for running the game and for providing the context of any tutorial intervention. The tutoring knowledge model includes teaching strategies and the game's goals and will be used in conjunction with the four standard operations. The student knowledge model includes a student-player's past and current roles and knowledge, missing conceptions and diagnosed misconceptions.

6.1. The Domain Model

Organising the domain knowledge in cards and stacks involves decomposing the domain knowledge into different hierarchical and non-hierarchical hypertext information cards whose level of domain detail depends on their position in the hierarchical structure [Angelides and Gibson, 1993] [Angelides and Tong, 1994]. The context of these cards is exclusively domain knowledge. It contains neither any knowledge about the student or what to do with this knowledge (i.e. tutoring knowledge). A card may be linked with any other card via an organisational hypertext information link, if there is a hierarchical relationship between the two cards, via a referential hypertext information link, if the two cards are non-hierarchically related, via a keyword hypertext information link if they share the same attribute, and finally via an annotation, if additional information about a card cannot be included in its context, for example graphs. By

being explicit, a hypertext information link carries a name which designates the relationship between the two cards it links.

The Domain Model includes all the knowledge about the "metal box" game. This should first include an initial scenario card (or a stack of cards) that introduce(s) the game by stating its aims and principles, rules, roles, sequence of action, and symbology. This is not intended as a replacement of the Player's Manual but as a complementary resource which the user may recall at any time. The Domain Model includes all knowledge about the four gamed roles, Managing, Financial, Production and Sales Director, as four separate cards which may be stored together in a roles stack. It should also include knowledge about the CDS, COS and Decision Slips, in three separate cards (but all stored in the same stack called notes), which may include, or may be linked to cards with, worked examples of these as well as blank cards that depict the CDS, COS and Decision Slips that the players will fill in during game play. Since the Production Director is responsible for filling in the CDS, the Finance Director the COS and the Managing Director the Decision Slip, there may be referential links from the Production Director's card to the CDS card, from the Finance Director's card to the COS card and from the Managing Director's card to the Decision Slip card.

The Domain Model should also include a stack of cards that depict the rules of the game. Thus it should include cards about production, production costs, the market, time sequence of production and sales and cash flow, market research, product sales, product advertising, R&D and finance. These cards may be used for game play or as a source of information for tutoring purposes. There may referential links from the rules cards to relevant cards in the roles cards and the CDS, COS, etc. cards. The Domain Model should also include an accounting system for performing various calculations. Finally, the Domain Model should also include a card or stack of cards that contain the steps of play.

In addition to using its roles, rules, steps of play and notes stacks for running the game, the Domain Model may use its knowledge about the game, for example, to explain the rules of the game should the environment detect a deviation from them, to offer some help in applying them correctly, to explain the purpose of a particular role, to correct incorrect role behaviour, etc. For this purpose, the Domain Model should have access to a library of all common misconceptions about the game.

6.2. The Tutoring Model

The Tutor Model includes knowledge about the game's teaching goals which it will try and help the student to attain through game play; thus it knows when, where and how to start, progress, and finish a game [Angelides and Gibson, 1993] [Angelides and Tong, 1994]. The Tutor Model supervises the flow of the game by controlling the steps of play, from introduction to the game through to postgame discussion or critique. In addition, by being in control of the time mechanism, it is in control of all the time parameters. All tutoring activities are under the control of computational teaching strategies which the Tutor Model uses to let the game flow. If a misconception is diagnosed for a player, then the Tutor Model goes into remedial mode for that player.

For every domain card there is a corresponding set of local teaching goals. Teaching goals are stored in cards. A teaching goals card contains attribute slots whose values are the local goals which the environment will try and help the student-player to attain during the course of interaction. Which teaching strategy will be applied is decided either by reference to the student model or by the user, during the course of interaction. A teaching strategy is an implementation of a particular teaching strategy used by human teachers, for example, coaching, questions/answering, evaluation of student responses, etc. This implementation contains tutoring knowledge about material presentation, for formulating tasks/responses to the student-user, for student evaluation and for remedial action. This implementation of a teaching strategy is stored in a "typed" card. A "typed" card is neither part of any form of hierarchy nor does it contain any hypertext information links to any other hypertext information cards.

As with the domain model cards, a teaching goals card is linked with organisational hypertext information links to other teaching goals cards. This sets up a hierarchical structure of

teaching goals cards. Since a teaching goals card is designated for a specific domain card, there can be no referential or keyword hypertext information links from a teaching goals card to another. However, there are annotations from each and every attribute slot (i.e. a goal) to all teaching strategies that are suitable for attaining a game goal denoted by a slot.

6.3. The Student Model

The Student Model includes the current knowledge of the player about the game, especially how well he mastered the rules of the game, the role he played during game play, his performance during the different steps of play and how well he managed the resources he was allocated (if any) by the Environment [Angelides and Gibson, 1993] [Angelides and Tong, 1994]. The student model also includes a record of all missing conceptions and misconceptions he has been diagnosed to suffer from, and whether these have been remedied at any stage, as well as any difficulties he experienced during the play. The Student Model also includes an indication of those teaching goals that the player has sufficiently demonstrated that he satisfied. The Student Model is a useful source of information during game play because it provides the basis on which the Environment can make decisions, such as further distribution resources, role reassignment, and game progress reports. The Student Model is an invaluable source of information for postplay evaluation. The feedback which the Student Model can provide can be used by the game designer for the development of initial player student models.

The student model is constructed during the course of interaction as an overlay model of the domain model, including diagnosed missing conceptions and misconceptions. For every domain card that the Environment uses with a student-player for game play or for tutoring, a corresponding student card is constructed, the contexts of which are determined by the context of the domain card. The attribute slots of the student card are a copy of the attribute slots of the corresponding domain card with the inclusion of some additional attribute slots to indicate the different paths that lead to the domain card (e.g. as part of the regular tutoring process or as part of some remedial action), how the domain card was used (e.g. to clarify the content of an attribute slot), and various teaching strategies that have been successfully or unsuccessfully applied with the domain card. The values in the attribute slots of the student card are obtained during the course of interaction.

Since student cards are constructed during the course of the tutoring process, any resulting hypertext information links are computed at the same time. Nevertheless, the end result will be, as with the domain model and the tutoring model, a network of student cards. A student card may be linked with organisational hypertext information links to other student cards. These organisational hypertext information links set up a hierarchical structure of student cards. These links, in addition to the hierarchical structure which they delineate, also define the overlay model of the student-user. There may be referential or keyword hypertext information links drawn from one student card to another, if the student-user establishes a non-hierarchical relationship between two domain cards or if two student cards share the same attribute slot value. These links may have an overlay statistic attached to the name which they carry. A numerical value is a standard yardstick of measurement in overlay models.

With every attribute slot there may be a set of associated misconceptions. Every time the student-user provides input which is not recognised by the Environment as correct, the Environment checks through the library of misconceptions for a known misconception. If this is the case, the Environment would record this as a student misconception in the attribute slot and create a referential link from it to the student card that describes the misconception. The name of the rule that proves that the answer is a known misconception is set as the name of the referential link. Finally, from each and every attribute slot that has been filled with a value there may be annotations to the best teaching strategy for acquiring the knowledge and achieving the goal and also to those teaching strategies that have been tried unsuccessfully.

6.4. The User Interface

The Hypertext Development Approach will yield an Integrated Environment with a Hypertext-orientated HCI which is supported by both the

Hypertext network structure of the Domain Knowledge representation and by the Integrated ITS [Angelides and Gibson, 1993] [Angelides and Tong, 1994]. Therefore, the interface to this Integrated Environment allows users to communicate both in first-person and second-person modes.

A Hypertext-oriented HCI supported by a Hypertext network structure allows a user to perform actions by manipulating icons that link directly to objects and actions in the Environment and link to other related objects and actions which is typical of a first-person interface mode. The representation of domain knowledge into semistructured hypertext information links and the resulting hypertext network structure requires the existence, and provides for the development, of a customised second-person interface mode in order to provide tutoring support effectively. This second-person interface mode may be in the form of a command language, menu or (even more ambitiously) natural language. Both the Hypertext Development Approach and the resulting accessibility of the Gaming-Simulation Environment through an HCI that supports both kinds of interface modes will either eradicate or reduce the constraints that will affect the human computer interaction with the HCI.

The semantically meaningful decomposition of the domain knowledge and its representation as semistructured hypertext information cards which are usually a screenful of domain knowledge each will help not to create any task constraints. The resulting networks of hypercards will impose a structure on the activities the user will be getting engaged in which in turn will impose a structure on the HCI in order to support this.

However, the hypertext base of HCI will also result in encouraging the user to explore as it is usually the case with such kinds of interfaces. The structure imposed by the resulting network of hypercards will also allow the Integrated ITS to provide help through the HCI in the form of Tutoring. The kind of help will depend on the implementation of the Integrated ITS.

A Gaming-Simulation will go through a cycle and follow certain steps of play. This assumes that there will be some sequence of events to be performed by the players or the Environment itself. At every phase of the cycle there is an attainable goal related with a single hypercard or group of hypercards. This could be stated in the HCI and the student's attention brought to it. During this cycle, the players go through different conceptual learning stages which they will connect their simulated experience to the real world.

The Integrated ITS is capable to provide tutoring at different competence levels and this feature combined with the availability of both kinds of interface modes will allow the HCI to provide access to the Gaming-Simulation Environment at varying levels of abstraction depending on the level of competence of the student and at a high degree of fidelity.

The Environment integrates Intelligent Tutoring and a simulation game. Consequently, the Environment may induce direct learning through tutoring about the domain knowledge of the game or indirect learning through the game activities. The Environment through its Integrated ITS can compensate for weaknesses in the users' cognitive abilities by directly or indirectly providing missing conceptions or remedying misconceptions and in general provide tutoring wherever is needed. The HCI will be capable of conveying any information relevant either to tutoring task or game task in either (or both) of the interface modes and at the appropriate level of abstraction.

The integrated ITS can influence the capabilities of the interface in order to ensure that the user is carrying out semantically acceptable actions. Consequently, the HCI becomes an instructional environment in which user misconceptions could be remedied. Any physical requirements (e.g. a physical device like the CDS) may be reduced by directly representing it on a first-person interface mode as it appears in real life, or by animating it.

7. Concluding Discussion

The HCI to the Integrated Environment defines the way that students think about the concepts in which they are being taught [Miller, 1998] [Burton, 1988]. Human-computer interaction in such terms is not a mechanical exchange of actions, but a communication of concepts, a semantic process in which the interface reflects the semantic nature of this interaction.

The interface needs to embody an understanding of, and appreciation for, the goals and concepts that are important to users both in the game and the inherent domain being tutored. Consequently, it needs to embody an understanding of the user's cognitive abilities and limitations, and the domain of the game to which the interface serves as a portal. Therefore, the important issue is not the application area of the interface but the definition of the ways in which good interfaces can support people as they gradually acquire an understanding of a complex semantic domain.

A Hypertext Approach in developing the Integrated Environment and consequently the HCI to this Gaming-Simulation Environment for Learning promises to relinquish or at least minimise the constraints that may affect human computer interaction with the Interface.

References

J. R. ANDERSON, The Expert Module. In *Foundations of Intelligent Tutoring Systems* (M. C. Polson and J. J. Richardson, Eds.), (1988) pp. 21–53. Lawrence Erlbaum, USA.

M. C. ANGELIDES, *Developing the Didactic Operations for Intelligent Tutoring Systems: A Synthesis of Artificial Intelligence and Hypertext*, PhD. Thesis, University of London, 1992.

M. C. ANGELIDES, Providing Intelligent Tutoring within a Gaming-Simulation Environment for Learning, submitted for publication in the Games, Simulations and Human-Computer Interfaces special issue of the *Journal of Simulation and Gaming* (1994).

M. C. ANGELIDES and G. GIBSON, Pedro — The Spanish Tutor: A Hypertext-based Intelligent Tutoring System for Foreign Language Learning, *Hypermedia*, 5(3) (1993)

M. C. ANGELIDES and A. K. Y. TONG, Implementing Multiple Tutoring Strategies in an Intelligent Tutoring System for Music Learning, *Journal of Information Technology*, 9(3) (1994).

M. C. ANGELIDES and R. J. PAUL, Towards a Framework for Integrating Intelligent Tutoring Systems and Gaming-Simulation. In *Proceedings of the 1993 Winter Simulation Conference* (G. W. Evans, M. Mollaghasemi, E. C. Russell and W. E. Biles, Eds.), (1993a) pp. 1281–1289. Los Angeles, USA.

M. C. ANGELIDES and R. J. PAUL, Developing an Intelligent Tutoring System for a Business Simulation Game, *Simulation Practice and Theory*, 1(3) (1993b), 109–135.

L. BIELAWSKI and R. LEWAND, Intelligent Systems Design: *Integrating Expert Systems, Hypermedia and Database Technologies*. John Wiley and Sons, USA, 1991.

R. R. BURTON, The Environment Module of Intelligent Tutoring Systems. In *Foundations of Intelligent Tutoring Systems* (M.C. POLSON and J.J. RICHARDSON, Eds.), (1988) pp. 109–142. Lawrence Erlbaum Associates, USA.

W. J. CLANCEY, *Knowledge-based Tutoring: The GUIDON Program*. MIT press, USA, 1987.

E. H. SHORTLIFFE, *Computer-based Medical Consultations: MYCIN, Artificial Intelligent Series*. Elsevier, New York, 1976.

J. R. MILLER, The Role of Human-Computer Interaction in Intelligent Tutoring Systems. In *Foundations of Intelligent Tutoring Systems* (M. C. Polson and J. J. Richardson, Eds.), (1988) pp. 143–189. Lawrence Erlbaum Associates, USA.

(CRAC) CAREERS RESEARCH AND ADVISORY CENTRE, *'Stelrad Limited' The Metal Box Business Game*, Cambridge: Hobsons Press, 1978.

J. NIELSEN, *Hypertext and Hypermedia*. Academic Press, USA, 1990a.

F. G. HALASAZ, Reflections on notecards: Seven issues for the next generation of hypermedia systems, *Communications of the ACM*, 31(7) (1988), 836–852.

H. M. HALFF, Curriculum and Instruction in Automated Tutors. In *Foundations of Intelligent Tutoring Systems* (M. C. Polson and J. J. Richardson, Eds.), (1988) pp. 79–108. Lawrence Erlbaum, USA.

J. NIELSEN, The Art of Navigating through Hypertext, *Communications of the Association of Computing Machinery*, 33(3) (1990b), 297–321.

A. F. SMEATON, Retrieving information from hypertext: Issues and problems, *European Journal of Information Systems*, 1(4) (1991), 239–247.

A. STEVENS, A. COLLINS and S. E. GOLDIN, Misconception in Students Understanding. In *Intelligent Tutoring Systems* (D. Sleeman and J. S. Brown, Eds.), (1982) pp. 13–24. Academic Press, USA.

K. VANLEHN, Student Modelling. In *Foundations of Intelligent Tutoring Systems* (M. C. Polson and J. J. Richardson, Eds.), (1988) pp. 55–78. Lawrence Erlbaum, USA.

Contact address:
Marios C. Angelides and Ka Yan Tong
Information Systems Department
London School of Economics
and Political Science
Houghton Street
London WC2A 2AE, U.K.

MARIOS C. ANGELIDES is a Lecturer in the Information Systems Department at the London School of Economics. He holds a B.Sc. degree in Computing and a Ph.D. in Information Systems, both from the London School of Economics. He has six years of experience in researching in the area of intelligent tutoring systems in which he completed his Ph.D. He has authored and co-authored twelve journal papers in intelligent tutoring systems. In addition he has published another six articles in other areas of artificial intelligence, one of which is in the area of artificial intelligence and simulation. He is the co-author of the book *Lisp: From Foundations to Applications* published in 1988. He is Vice-Chairman of IFIP's (International Federation for Information Processing) Working Group 9.5: Social Implications of Artificial Intelligence Systems.

KA YAN TONG is a full-time research student reading towards the degree of Ph.D. (Econ) in Information Systems in the Information Systems Departmentat at the London School of Economics. Her research area is that of intelligent tutoring systems. She holds the degrees of B.Sc. in Computing and M.Sc. in Analysis, Design and Management of Information Systems, both from the London School of Economics.