A Formal Tutoring Process Model for Intelligent Tutoring Systems

Alke Martens and Adelinde M. Uhrmacher[1]

Abstract. The combination Computer Based Training systems with Artificial Intelligence and Cognitive Science has led to the development of Intelligent Tutoring Systems nearly 30 years ago. A common agreement has been reached about the constituents of an Intelligent Tutoring System (ITS). Nonetheless, the interpretation of the role of each component in the ITS is still heterogeneous. Together with the absence of formal methods in ITS, this leads to the situation, that the components of ITSs are strongly domain dependent and not reusable. The situation is analyzed for case-based ITS, i.e. ITS based on a narrative story line. A formal model of the tutoring process is introduced into the ITS architecture. The model is based on the idea to support the training of two cognitive processes, i.e. the process of diagnostic reasoning, and the process of general knowledge application. The integration of the tutoring process model as the central component in the ITS has led to a homogenization of the architecture. Based on the new architecture, the ITS Docs 'n Drugs has been realized.

1 Introduction

Intelligent Tutoring Systems can look back on a tradition of nearly 30 years. In the 1970s, Carbonell [1] made the first attempts to combine Computer Aided Instruction systems (CAI) with Artificial Intelligence (AI). Reflecting the development of techniques and methods of AI, many different kinds of systems have been constructed in these years. Facets have been and still are development of dialogue techniques, planning, embedding agents as part of an Intelligent Tutoring System (ITS) or even constructing an ITS as a multi-agent system (for references see e.g. [6]). Simulation is part of many ITSs. ITSs have been designed to be used in the World Wide Web.

Nowadays, there exists a kind of common agreement about the ITS architecture. One of the first descriptions of this architecture can be found in [2], where Clancey described the ITS GUIDON and induces a model for the ITS architecture. Others have followed and refined this description to what today can be called the classical ITS architecture (see amongst others [3, 4, 11, 12]). The classical ITS architecture consists of four models: the expert knowledge model, the pedagogical knowledge model, the learner model, and the user interface. The naming of the components varies (see e.g. [4] in comparison with [3]). Some systems have an additional exercise generator embedded (e.g. [4]). However, an exercise generator can not be used in case-based training as described in this paper, so it will be left out in the following.

Regarding the classical ITS architecture, it should be easy to reuse components of existing ITSs for constructing a new ITS. However, the reuse of components is seldom possible. One reason can be seen in the fact that most of the ITSs are domain dependent. Moreover, investigation and comparison of descriptions of ITS architectures have shown that the interpretation of the role of each component in the ITS varies a lot (for references and detailed description of this research see [6, 5]). Some systems centralize the expert knowledge model and use it as central expert system (e.g. [2]). Some others use the expert knowledge model as a complex database and focus on the pedagogical knowledge model (e.g. [8]). The heterogeneous interpretation of the role of each component in the ITS, in combination with the tendency to ad hoc implementation in the area of ITS development, has led to incomparable and not reusable systems.

The approach described in the following has its background in the development of the web- and case-based ITS 'Docs 'n Drugs - the Virtual Policlinic' [7] (see www.docs-n-drugs.de). Docs 'n Drugs is part of the medical curriculum at the University of Ulm since the year 2000. Several hundreds of students have worked with the system, yet.

To develop Docs 'n Drugs, the classical ITS architecture and its applicability to case-based training has been analyzed. One result has been the insight, that a reuse of existing ITS or of ITS's components is not possible. Thus, the ITS architecture has been adapted to the requirements of case-based training. A side effect has been an approach towards homogenization of the ITS architecture, as described in [5]. The suggested new architecture comprises all components of the classical ITS architecture, as mentioned above. However, expert knowledge model and pedagogical knowledge model are reduced to databases without own execution and delivery. Execution, such as steering of the interaction with the learner, and delivery of contents must be taken over by another component. Thus, in addition a new formally described component has been embedded as the central component in the ITS. This component is called the tutoring process model. The advantage of such an approach is the clear separation of domain dependent content and the domain independent delivery. This might be a basis for the development of exchangeable components. Moreover, the formal model of this central component provides an implementation independent description.

In the following, a short introduction in case-based training will be given. After that, the tutoring process model is presented. The tutoring process model is constructed as an abstract model with two extensions, i.e. the basic tutoring process model and the adaptive tutoring process model [6]. The latter, which is focus of the paper, embeds a learner model. The paper closes with a conclusion and an outlook.

2 Case-based Learning

Case-based learning has a long tradition in some areas, e.g. law, medicine, and business. Case-based training has often been realized

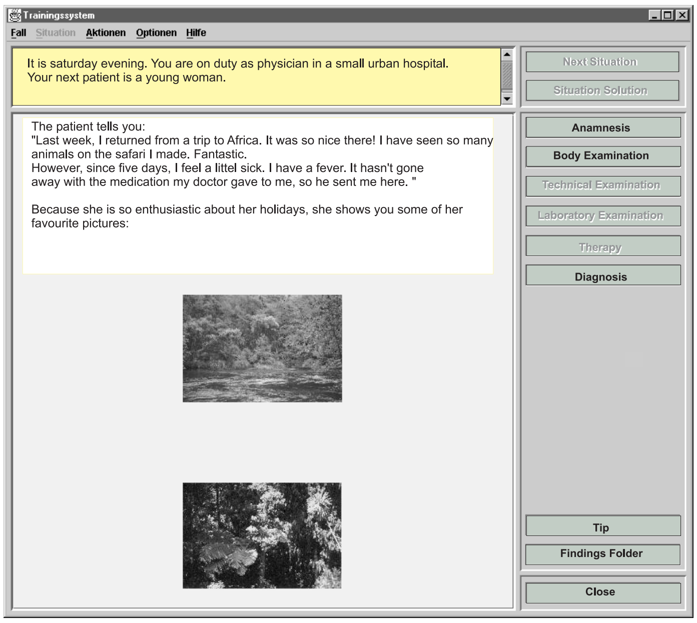
[1] University of Rostock, Institute for Computer Science, Rostock, Germany email: martens@informatik.uni-rostock.de

in ITS. Often problem-based learning has been mixed up with case-based learning. Whereas case-based learning is based on a narrative story line which contains a series of problems, problem-based learning is based on a problem description. In case-based learning, the story's development is influenced directly by the learner's decisions. In problem-based learning, the problem description does not change and does not develop.

Docs 'n Drugs helps the students to train everyday clinical practice in a case-based manner. The student takes over the role of a physician in a hospital. He is confronted with a patient and has to choose the correct and appropriate sequence of steps to treat the patient. If he chooses wrong steps the patient's state might deteriorate. Additionally, the training cases are enriched with interaction parts, where the student has e.g. to answer questions. The approach chosen in Docs 'n Drugs is based on the idea of supporting two cognitive processes in case-based training. This idea is based on research about cognitive processes in everyday clinical practices, described in [9, 10]. One insight of this research has been that in everyday clinical practice at least two cognitive processes are distinguishable:

1. The process of general knowledge application, such as knowing how to act and react in an appropriate manner, and how to choose the correct methods at the right time.
2. The process of diagnostic reasoning, such as taking the information given and deducing a correct diagnosis.

Seemingly, these two processes are not dependent of the application domain. The assumption is that case-based training should support training of the two cognitive processes, independent of the training domain.



It is saturday evening. You are on duty as physician in a small urban hospital.
Your next patient is a young woman.

The patient tells you:
"Last week, I returned from a trip to Africa. It was so nice there! I have seen so many animals on the safari I made. Fantastic.
However, since five days, I feel a littel sick. I have a fever. It hasn't gone away with the medication my doctor gave to me, so he sent me here. "

Because she is so enthusiastic about her holidays, she shows you some of her favourite pictures:

Figure 1. Screenshot of Docs 'n Drugs

In Docs 'n Drugs, the screen shows information about the patient, for example results of examinations, and utterances of the patient. Page contents are adaptable to the student's current performance and his profile such as his experience in the application domain. The navigation shows typical steps of the patient treatment process. Figure 1 shows a screenshot of the ITS. The navigation consists of the so-called actions. Actions can be distinguished in temporarily available actions and permanently available actions. Whereas the number of temporarily available actions is adapted to the learner at run time, the permanently available actions are always active. One of the permanently available actions is the differential diagnosis. The student should note his suspected diagnoses continuously during the interaction with the training case. The correction of this list of diagnoses takes place based on the facts and rules in the expert knowledge model and takes into account what the student has learned about the case so far. In continuously actualizing and correcting his list of suspected diagnoses, the student is training the process of diagnostic reasoning.

Via choosing one of the temporarily available actions, the student is training the process of general knowledge application, such as 'which is the appropriate next investigation, given the patient's utterances and the results of the examinations'. The contents and information displayed should adapt to the student's performance and his chosen sequence of steps. For example, a repetition of an anamnestical question might lead to another answer. Three kinds of training cases can be distinguished:

1. Guided training cases - the student has only one action available. Thus, this kind of training case offers no possibility to choose next steps. It is suitable for beginners in the application domain or for case presentation. The process of general knowledge application can only be trained in embedded interaction elements.
2. Half-guided training cases - the amount of temporarily available actions is adapted at run time, taking the student's performance and knowledge into account. These training cases can be realized in different granularities and for students with different expertise. The adaptive tutoring process model plays its main role in steering and adapting this kind of training case.
3. Unguided training cases - always all temporarily available actions are active. Thus, no adaptation takes place. This kind of training case is only suitable for experts in the application domain.

The following section focuses on the half-guided training cases.

3 Tutoring Process Model

The tutoring process model is the new central component of the ITS. Related with the expert knowledge model, the tutoring process model has access to the relations and entities of the domain's expert knowledge. In case-based training, the pedagogical knowledge can be perceived as knowledge about the case's contents, i.e. the didactically elaborated elements of the training case. This knowledge may differ from the expert knowledge such as it might contain misleading or incorrect information. In Docs 'n Drugs, each entry in the pedagogical knowledge is related to entries in the expert knowledge base. This relation is used by the tutoring process to draw conclusions, to construct corrections or help, and to adapt the training case. The tutoring process model is also related with the learner model. Thus, it has access to all information in the learner model, such as the learner's profile and the learner's performance in the training case.

As a formal model, the tutoring process model is described as the so called 'abstract tutoring process model' with two extensions: basic tutoring process model and adaptive tutoring process model. The abstract tutoring model consists of the training case C and additional functions:

$$TPM = \langle C, show, enable \rangle \qquad (1)$$

Here, C is the training case, $show$ is the show state function, and $enable$ is the enable action function.

The basic tutoring process model is structured the same way. It allows the construction and steering of training cases without an existing learner model. Thus, no adaptation to the learner model can take place in the basic tutoring process model.

The adaptive tutoring process model contains a learner model. Thus, it is a structure:

$$TPM_{adapt} = \langle C, LM, show, enable \rangle \tag{2}$$

Here LM is the learner model.

In the following text, firstly the learner model is described, followed by a description of the adaptive training case. The section closes with the description of the adaptation functions of the adaptive tutoring process model.

3.1 Learner Model

The learner model provides the basis of the adaptation. Thus, it should contain at least two types of information. The first type is information about the learner such as the learner's profile, his level of experience, and his identification. The second type is information about the assumed knowledge of the learner, e.g. a history of his chosen sequence of steps.

Accordingly, the learner model can be described as a structure consisting of a learner profile LP and the knowledge of the learner LW:

$$LM = \langle LP, LW \rangle \tag{3}$$

Here, LP is a structure $LP = \langle id, expertise \rangle$, and LW is a structure $LW = \langle id, Lpath, Lresult, Lacq \rangle$

LP's components are: id - the identification of the learner, and $expertise$ - the learner's expertise.

LW's components are: id - the identification of the training case, $Lpath$ - the partial path, $Lresult$ is a set of results, and $Lacq$ is the set of facts.

The learner model LM currently defines a minimum of information required for the adaptation process. The described prototype [6] does not perform an actualization of the learner profile during the interaction of a learner with a particular training case. In contrast to this, the learner's knowledge must be actualized at run time. Thus, the information about the path chosen by the learner, the results achieved, and information about the collected facts are updated after every step of the learner.

The partial path in $Lpath$ denotes the sequence of actions chosen by the learner (for definition of paths and partial paths see [6]). Thus, it directly reflects one part of the process of general knowledge application. After the learner has selected a subsequent step, this partial path is actualized. The second part of the process of general knowledge application is recorded in the set $Lresult$, showing the results of the learner's interaction with the interaction elements in the training case. The set of facts $Lacq$ reflects the amount of information which has de facto been displayed to the learner. After selecting the information elements that should form the actual display, the facts related to each of the elements are recorded in $Lacq$.

3.2 Training Case

The training case C is a structure:

$$C = \langle Q, A, q_0, F, B, \delta, select, allow \rangle \tag{4}$$

with:

Q	finite set of states
A	finite set of actions
$q_0 \in Q$	start state
$F \subset Q$	finite set of final states
B	finite set of bricks
δ	state transition function
$select$	select brick function
$allow$	select action function

The description shows that each training case consists of a set of states Q and actions A. Each state consists of a set of bricks B. A subset of bricks forms the display of the learner (see figure 1). Bricks exist as information bricks, containing only multimedia information, and as interaction bricks, containing interaction elements, e.g. multiple-choice elements. Each action A is associated with a state. If the learner chooses the action he will reach the according state. Actions are the temporarily available actions, mentioned in section 2. These actions form a part of the navigation in figure 1.

The state transition function δ of the training case C is constructed as

$$\delta : Q \times A \longrightarrow Q \cup \{\perp\} \tag{5}$$

For each state (including the start state) and each action, the state transition function defines the subsequent state. If the application of the state transition function on a state $q_i \in Q$ and an action $a \in A$ leads to a state q_i', then the application of the state transition function on another state $q_j \in Q$ with the same action a must lead to the same state, i.e.:

$\forall q_i, q_j, q_i', q_j' \in Q \backslash F, a_k \in A :$

$\delta(q_i, a_k) = q_i' \wedge \delta(q_j, a_k) = q_j' \Rightarrow q_i' = q_j'$

The symbol $\perp$ denotes, that the actual implementation of the system should prevent this state transition to be executed. It is:

$\forall q_f \in F, F \subset Q, a \in A : \delta(q_f, a) = \perp$

The function $select$ is responsible for selecting all the bricks that are part of a state:

$$select : Q \longrightarrow 2^B \tag{6}$$

It is: $\forall q \in Q : select(q) \neq \emptyset$. Thus, each state must have at least one brick. This is one way to make sure that states are not empty - and that the learner does not get an empty display.

The function $allow$ determines all the actions which are associated with one state:

$$allow : Q \longrightarrow 2^A \tag{7}$$

It is: $\forall a \in A, q \in Q : a \notin allow(q) \Rightarrow \delta(q, a) = \perp$

Thus, if an action is not part of the set of actions determined by $allow$, the state transition function δ, applying this action to any given state, will lead to the result $\perp$. If the state $q \in Q$ is a final state, i.e. $q_f \in F$, then the $allow$ function will return an empty set: $q_f \in F : allow(q_f) = \emptyset$. Otherwise, the set of actions determined by $allow$ mustn't be empty: $\forall q \in Q \backslash F : allow(q) \neq \emptyset$

3.3 Structured Bricks and Actions

In the adaptive tutoring process model, bricks and actions are structures themselves. A brick $b \in B$ is a structure:

$$b = \langle id, con, PRE, POST \rangle \tag{8}$$

with: id is the identification of the brick, con is the list of content elements, PRE is the set of pre conditions, and $POST$ is the set of post conditions. The con list can be distinguished in con_{info}, i.e. a multimedia information element, and $con_{interact}$, i.e. a multimedia interaction element. The post condition $POST$ denotes the facts of the training case that reflect the state of the training case after the brick has become part of the page to be displayed to the learner. Thus, the facts noted in the bricks that have been displayed are added to the set $Lacq$ of the learner model.

An action $a \in A$ is a structure:

$$a = \langle id, name, PRE \rangle \qquad (9)$$

with: id is the identification of the action, $name$ is the (display) name, and PRE is the set of pre conditions. The pre condition PRE determines, whether the according brick or action will be part of the actual display of the current learner. The evaluation of a pre condition takes into account the learner's profile LP as well as the learner's knowledge LW. The other way around, pre conditions can be constructed by the training case authors using as well profile information ('this brick should only be seen by experts') as facts in the training case ('if fact XY has been seen, the action Z should be active').

3.4 Adaptation

The functions $show$ and $enable$ in the adaptive tutoring process model are mainly responsible for the adaptation of states and actions. The show state function $show$ is a function

$$show : 2^B \times LP \times LW \longrightarrow 2^B \qquad (10)$$

with: $show(B_q, expertise, Lacq) = B_a$
B_q is the set of bricks determined by $select$, i.e. the bricks associated with the state Q. $expertise$ is the profile entry of the learner model, and $Lacq$ is the amount of facts the learner has acquired so far in the training case. The function $show$ determines, which subset of bricks will actually be displayed, taking the learner model into account, with $B_a \neq \emptyset$, i.e. there should be at least one brick that can be shown to the learner. With this condition, the tutoring process ensures that no empty display occurs accidentally. Thus, with $b \in B$ and $pre_b \in PRE_b$ it is:
$(\, B_a \subseteq B_q \subseteq B \,) \wedge \forall b \in B_a \,.$
$(\, pre_b \in Lacq \vee pre_b = expertise \vee PRE_b = \emptyset \,)$
The set of pre conditions of a brick can be the empty set - this kind of brick will be displayed without condition to every type of learner.

The enable action function $enable$ is a function

$$enable : 2^A \times LP \times LW \longrightarrow 2^A \qquad (11)$$

with: $enable(A_q, expertise, Lacq) = A_a$
Similar to the $show$ function, the $enable$ function takes the set of actions determined by $allow$, i.e. A_q, and derives which actions will actually be available in this step, i.e. A_a with $A_a \neq \emptyset$. It uses the entries $expertise$ and $Lacq$ in the learner model. So, the set of temporarily available actions is adapted to the learning at run time. However, there should be at least one action that meets the pre conditions, i.e. with $a \in A_a$ and $pre_a \in PRE_a$ it is:
$(\, A_a \subseteq A_q \subseteq A \,) \wedge \forall a \in A_a \,.$
$(\, pre_a \in Lacq \vee pre_a = expertise \vee PRE_a = \emptyset \,)$
As can be seen above, the set of pre conditions PRE of an action $a \in A$ can also be the empty set - then, the action is always active.

With all the parts specified in the subsections, the adaptive tutoring process model is a structure:
$TPM_{adapt} = \langle Q, A, q_0, F, B, \delta, select, allow, LP, LW, show, enable \rangle$

3.5 Sequence of Functions in Adaptation

The sequence of functions to be executed is important. The sequence after the selection of an action by the learner is:

- Determine the next state with the δ function.
- Take this state and derive the set of associated bricks, using the $select$ function.
- Take this set of bricks and the learner model and determine with $show$, which of the bricks should be shown to the current learner.
- Actualize the learner model with the bricks' post condition (only of the bricks determined for display).
- Determine the amount of actions associated with the state, using the $allow$ function.
- Take this set of actions and the actualized learner model and determine with $enable$ which actions should be available to the current learner.

This sequence is important, because after determining the bricks that should be displayed, the entries in the learner model are updated. This will likely influence the choice of available actions. The underlying assumption is that the learner will have the information of the bricks available and thus has a new state of knowledge - which should be taken into account when allowing him to select a subsequent step in the treatment process.

4 Example

In this section, an example of the adaptation of a state to a learner model will be given. The learner with the $id = 2$ works with the training case number 9. This training case tells the story of a young female patient, returning from a trip to Africa with a fever. The case is developed for beginners, advanced, and expert learners. Learner number 2 is a beginner, he is student of the second semester clinical medicine. The learner profile contains the following entries: $LP = \langle 2, 'beginner' \rangle$. The learner has seen the start page and has read the introduction to the training case. He has chosen the 'travel anamnesis' as the first action. Currently, he has the display 'travel anamnesis' available and the menu-buttons 'anamnesis' and 'body examination' (see figure 1). His learner knowledge has the entries:
$LW = \langle 9, (q_0, a_{travel}), \emptyset, \{\text{'sun-tanned skin', 'fever', 'africa'}\} \rangle$
As a next step, the learner clicks the anamnesis button. From the menu, he selects the 'inoculation anamnesis' $a_{ino} \in A$. He wants to see whether the patient has got the correct preparation for the trip to Africa.

After the selection, the tutoring process model's function δ determines the new state: $\delta(q_{travel}, a_{ino}) = q_{ino}$. The state $q_{ino} \in Q$ consists of four bricks, i.e. $select(q_{ino}) = \{b_1, b_2, b_3, b_4\}$. Brick b_1 is suitable for beginners, brick b_2 has an empty pre condition, brick b_3 is designed for experts, and to get brick b_4, the fact 'headache' must exist in LW. Taking the learner model's entries into account, the function $show$ determines the bricks b_1 and b_2 to be displayed:
$show(\{b_1, b_2, b_3, b_4\}, 'beginner', \{\text{'sun-tanned skin', 'fever', 'africa'}\}) = \{b_1, b_2\}$.
The following table shows the bricks. Now the new display and the new navigation can be shown to the learner.

brick	id	con	PRE	$POST$
b_1	1	($text$, 'The patient made malaria prophylaxis with the medication noted in her certificate of vaccination.')	'beginner'	'malaria prophy-laxis'
b_2	2	($picture$, vaccination card malaria)	$\emptyset$	$\emptyset$
b_3	3	($text$, 'The patient smiles and says: 'I have made malaria prophylaxis'.')	'expert'	$\emptyset$
b_4	4	($text$, 'The vaccination card tells you that the patient has made a malaria prophylaxis. Try to find out, if the medication noted down in the card might have effected the headache.')	'headache'	'malaria prophy-laxis'

After having decided, which bricks can be displayed, the tutoring process updates the learner model. When the learner has the derived display available, he has the new entry 'malaria prophylaxis' in his set of acquired facts $Lacq$ in the learner knowledge LW. With the updated knowledge in the learner model, the tutoring process now determines which actions should be available from the current state.

The function $allow$ takes the state q_{ino} and determines the set of actions associated with it: $allow(q_{ino}) = \{a_{ana}, a_{th}a_{lab}\}$. Here the example is shortened due to comprehensibility. Usually, the amount of available actions is larger. This set of actions is now used by the $enable$ function. This function derives, together with the actualized entries in the learner model, which actions will be available: $enable(\{a_{ana}, a_{th}, a_{lab}\},' beginner', \{$'sun-tanned skin', 'fever', 'africa', 'malaria prophylaxis'$\}) = \{a_{ana}, a_{lab}\}$.

The action a_{ana} is the first anamnesis. This action has been available in the beginning of the training case and has no pre conditions. Thus, it should be always available. The action a_{th} is an action allowing the learner to choose a therapy. Because the learner is a beginner, and has not acquired the necessary knowledge yet, this action is blocked. However, the action a_{lab}, leading to laboratory examinations, is available now, because the pre condition, i.e. the knowledge of 'malaria prophylaxis', is part of $Lacq$. The following table shows the actions.

action	id	name	PRE
a_{ana}	1	'Anamnesis'	$\emptyset$
a_{th}	2	'Therapy'	'expert'
a_{lab}	3	'Laboratory examination'	'malaria prophylaxis'

5 Conclusion and Outlook

In this paper a formal adaptive tutoring process model for case-based ITSs has been introduced. The tutoring process model provides a new perspective on the ITS architecture. It is used to separate content and delivery in the classical ITS architecture and thus provides the means for reusability of components in ITS.

The adaptive tutoring process model has been described in detail and the adaptation process has been sketched. The central tutoring process model of the ITS Docs 'n Drugs has been modelled based on the formal model. Thus, the tutoring process model prooved its viability and applicability.

The tutoring process model has been developed as an abstract model with the two extensions basic tutoring process model and adaptive tutoring process model. Whereas the first one does not need a learner model, the second one is described with an inherent, simple learner model. It should be easy and will be part of future work, to develop an extensive learner model and integrate it into the adaptive tutoring process model. The gain for the tutoring process model will be a finer grained adaptation process. For example, a finer distinction of the learner's expertise and its adaptation at run time would allow a better adaptation of the training case - a learner which starts as 'beginner' and performs very well, might ascent an become 'expert' at run time.

The tutoring process model can not only be used as the basis of the development of an ITS's central component, but also as a central component of an authoring system. There, the model can help to develop the training cases and simulate a learner's interaction with a training case. The interaction bricks of the tutoring process model haven't been specified yet. Here, future work could be to categorize different types of interaction and provide the authors with pre-implemented default interaction elements, where they only have to fill in content.

REFERENCES

[1] J. R. Carbonell, 'AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction', *IEEE Transactions on Man-Machine Systems*, **11**(4), 190–202, (1970).

[2] W. J. Clancey, 'Methodology for Building an Intelligent Tutoring System', in *Methods and Tactics in Cognitive Science*, eds., W. Kintsch, J. R. Miller, and P. G. Polson, 51–84, Lawrence Erlbaum Associates, (1984).

[3] A. T. Corbett, K. R. Koedinger, and J. R. Anderson, 'Intelligent Tutoring Systems', in *Handbook of Human-Computer Interaction*, eds., M. Helander, T. K. Landauer, and P. Prabhu, 849–874, Elsevier Science B.V., (1997).

[4] R. Lelouche, 'Intelligent Tutoring Systems from Birth to Now', *KI - Kuenstliche Intelligenz*, **4**, 5–11, (1999).

[5] A. Martens, 'Discussing the ITS architecture', in *Workshop 'Expressive Media and Intelligent Tools for Learning', 26th German Conference on AI KI-2003*, Hamburg, Germany, (2003).

[6] A. Martens, *Ein Tutoring Prozess Modell fuer fallbasierte Intelligente Tutoring Systeme*, Ph.D. dissertation, Rostock, Germany, 2003.

[7] A. Martens, J. Bernauer, T. Illmann, and A. Seitz, "'Docs 'n Drugs - The Virtual Polyclinic"', in *Proc. of the American Medical Informatics Conference AMIA 01*, Washington, USA, (2001).

[8] M. Mayo, A. Mitrovic, and J. McKenzie, 'CAPIT: An Intelligent Tutoring System for Capitalization and Punctuation', in *Proc. of the Int. Workshop for Advanced Learning Technologies IWALT 2000*, pp. 151–157, Palmerston North, (2000).

[9] V. L. Patel, D. R. Kaufman, and J. F. Arocha, 'Steering through the murky waters of a scientific conflict: situated and symbolic models of clinical condition', *AI in Medicine*, **7**, 413–438, (1995).

[10] V. L. Patel and A. W. Kushniruk, 'Understanding, Navigating and Communicating Knowledge: Issues and Challenges', *Methods on Information in Medicine*, 460–470, (1998).

[11] S. P. Smith, *Developing an authoring environment for procedural task tutoring systems*, Ph.D. dissertation, Massey University, Palmerston North, New Zealand, 1997.

[12] C. W. Woo, *Instructional Planning in an Intelligent Tutoring System*, Ph.D. dissertation, Dept. of CSAM, Illinois Institute of Technology, Chicago, IL, 1991.