

Applying learner modelling for user interface assistance in simulative training systems

Alexander Hörnlein, Frank Puppe

Dept. for Artificial Intelligence and Applied Computer Science, University of Würzburg
{hoernlein, puppe}@informatik.uni-wuerzburg.de

Abstract: Simulative case-based training systems in medicine should enable the user to act as a real doctor treating a real patient. This leads to a higher complexity of the user interface (UI), which in turn leads to a higher effort for the user to learn the handling of the UI. In this paper, we present how the adaptive help agent in the training system d3web.Train uses learner modelling to decide whether and what help the learner needs, so she can concentrate on the learning subject instead of the UI.

1 Introduction

d3web.Train [d3web.Train, HBP02] is an intelligent case-based training system in the web with the focus on giving the learner the possibility to play the role of a remote doctor acting on an electronic patient record (EPR). To achieve this d3web.Train enables the learner to do as many of the actions available for a real doctor as reasonable. However, this inevitably lead to a user interface (see Fig.1) with high complexity which is especially counterproductive in training systems, where the user already struggles to understand the learning subject. Recent evaluations [RKT04, HRKBP04] have shown that the user interface ergonomics were the most criticised issue while the overall rating of the system was very good. Some of the learners participating in the evaluation even complained that they missed one of the assessed tasks entirely, which lead to a bad overall assessment.

There are traditional ways to provide help to assist the user with the UI like introductory courses, built-in manuals, etc. (some of them are used with d3web.Train) - but they all lack a solution to the following issues:

- the help does not match the learner's knowledge
- the learner may not even know that she needs help

Thus we implemented a help module that tracks the learner's actions, models her understanding of the concepts that must be understood for successfully handling the system, and intervenes by displaying appropriate information as soon as it concludes that help is needed.

We restrict this paper to model the learner's abilities as far as the operating the UI is concerned and will not consider her understanding of the learning subject.

The paper is organized as follows: Section 2 presents the system and particularly its task domain. Section 3 explains the technique of overlay modelling, how we apply it to our system and provide necessary definitions. In section 4 we define the concepts about the

UI and formulate the model of the learner's mastery of these concepts and section 5 addresses the issue of appropriate intervention. Section 6 closes with a short discussion and future work.

2 d3web.Train and its task domain

Depending on the information available in the underlying case d3web.Train presents it is possible/necessary for the learner to perform four different main tasks:

- order examinations: The learner has to choose which of the available examinations has the highest ratio of information gain through cost and risk
- interpret results: If the examinations did not yield formal results but results of some other form (e.g. pictures like a radiograph) the learner has to determine the formal results for herself
- diagnose: Based on the results the learner has to decide which diagnoses are *suspected*¹ and which are *confirmed*.
- choose treatment: As soon as diagnoses are confirmed an appropriate treatment must be chosen

The case may then be presented again to the learner with the outcome of the (correct) treatment and the learner may again perform all these tasks until the case is finished. The typical workflow while working on a case in d3web.Train is depicted in Fig.1. The arrows inside the dotted session denote possible reasonable orders of tasks for the learner. However, the learner is not bound to follow this workflow during a session.

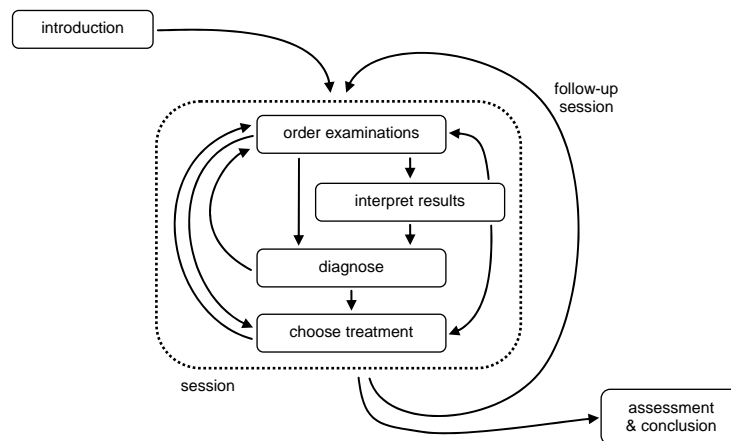


Figure 1: d3web.Train tasks

¹ There are only two possible ratings for diagnoses (and treatments) in d3web.Train: confirmed resp. indicated or suspected resp. reasonable.

Each task comprises either multiple partially ordered atomic actions or (sub-)tasks. As d3web.Train has an HTML based user interface, atomic actions are either clicks on sensitive areas or entering text. Other actions like “drag and drop” and further complicated actions (although possible with DHTML techniques) are not used. The general architecture of d3web.Train follows the *model view controls* paradigm [R79], where each atomic action (click of the user) is mapped onto an associated internal action which has a time-stamp and may have parameters.

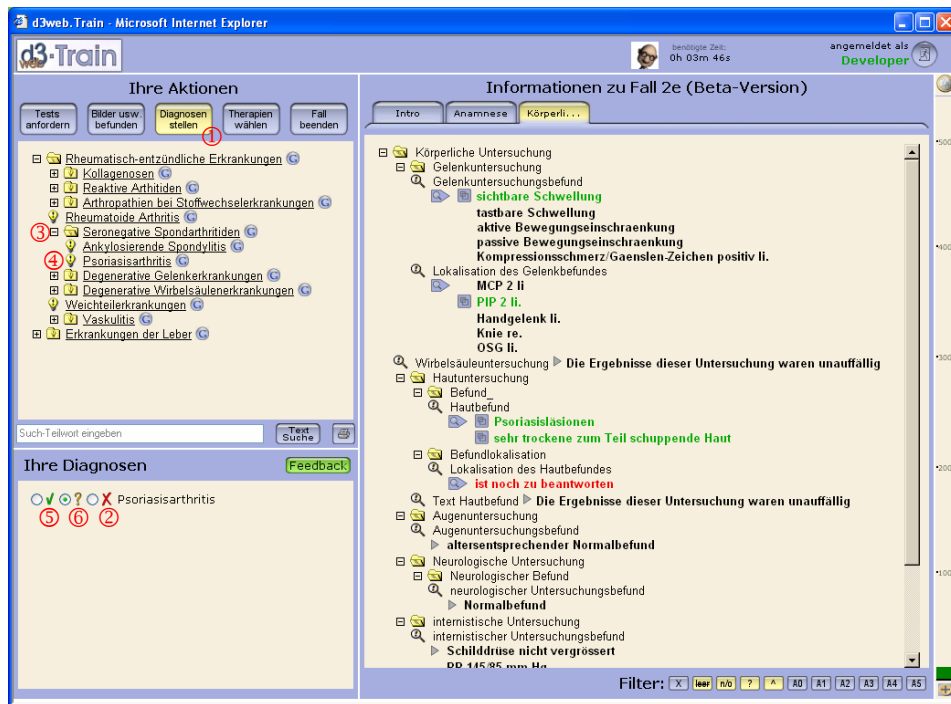


Figure 2: Screenshot of d3web.Train in diagnose mode.

Example

The task “diagnose” (Fig. 2) is made up of the following (sub-)tasks (t) and actions (a):



- (t) switch to diagnose mode by (a) clicking on the button “Diagnose” (1)
- (t) delete diagnoses which may neither rated as *confirmed* nor rated as *suspected* anymore
 - (a) click the delete button behind each offending diagnosis (2)
- (t) navigate the diagnoses tree by
 - (a) scroll to the sub-tree containing the diagnosis that is to be considered
 - (a) if not open: open this sub-tree (3)
 - (a) if too much sub-trees are open: close interfering sub-trees



- (t) add the diagnosis to the diagnoses notepad by (a) clicking on it (4)
- (t) rate the newly added diagnosis
 - (a) click on the radio button behind the symbol for *established* (5) or
 - (a) click on the radio button behind the symbol for *suggested* (6)

According to [N01] it is extremely important that the learner's mental model of the system matches the designer's (expert-)model. To achieve this, the system image (how the system is perceived by the learner) must be consistent with the designer's model.




To bias the learner's mental model initially into the right direction, the learner gets the following introductory explanation of the system metaphor:

Imagine being a doctor, who performs a consultation based on an electronic patient record. The EPR initially contains the history and physical examination only and is updated as soon as you order further examinations. Your job is to order examinations, interpret the results, diagnose and choose the correct treatment.


The EPR on the right side is divided in sections available by a click on the associated slider (anamnesis, physical examination, etc.). Graphical examination results are available with a click on  right at the examination or as an overview with  on the top left.

You have to choose suspected diagnoses based on the examinations and you have to choose examinations to eliminate or establish your suspected diagnoses. You can choose diagnoses with the diagnoses tree on the left side, examinations with the examinations tree and therapies with the therapies tree. Navigate trees with  and , and choose with a click on the name.

To switch between the trees use the buttons above the tree.

Diagnoses and therapies are added to your diagnoses and therapies notepads. There you have to rate the diagnoses and therapies with the radio buttons  as "established" and "indicated" or  as "suspected" and "reasonable". With  you can eliminate a diagnosis or therapy.

If you think your current diagnoses and therapies are correct you may finish the case.

Available additional background material for an element is indicated by a clickable .

To be able to rate the learner's abilities we first must make up our mind about the errors the learner may do and how we detect them.

3 Overlay modelling & system model

Adaptive help is only possible with a knowledgeable system that models the learner's mastery of the UI [FLS85]. A general overview over different methods can be found in [J03], an extensive work that concentrates more on physiologically motivated models is [Y03]. We choose overlay modelling [CG77] to acquire the learner's abilities concerning the UI as the most promising while feasible approach.

Usually overlay modelling is used to rate hypotheses about the learner's declarative knowledge of the domain, e.g. her understanding of diagnoses or even relations (rules) on a lower granularity level. In contrast, our overlay model is related to tasks and actions: the learner must know when she should solve a task and what actions are necessary for solving it. If the learner does not have this knowledge, she would either miss

some tasks at all in a case, or the actions for solving a task are incomplete and therefore suboptimal. We illustrate this with examples from Fig. 1 and 2: Missing a task can be detected quite easily with rules like:

- If a question is presented with the text “ist noch zu bearbeiten” (“needs to be handled”), and the learner does not click on this text (and has never clicked on such a text in the case till now), then she has missed the task “interpret results”.
- If the learner orders tests without having entered any diagnoses yet, he might have missed the task “diagnose” (this is not a categorical rule).

It is a little more difficult to recognize, whether the learner knows the action sequence to solve a task, because many actions are optional and depend on the goal the learner is pursuing. For example, if the learner does not know how to open a sub-tree, she will choose only top-level diagnoses or if she does not know how to rate a diagnoses, she will always stick to the default rating given by the system (initially “suspected”, later “established”). However, this may be done in some situations intentionally. Therefore, the system needs to combine hints, e.g.

- How often did a user miss a type of action, where the action was appropriate according to the correct solution of the current task?
- Did the learner perform a similar action type for another task (e.g. open a sub-tree for test-selection)?

In the following, we describe more formally, how the overlay model is represented.

Definitions of tasks and actions

Based on [Ma03] we call A the set of all available atomic actions and Q the set of all possible *partial* system states with the transition function $\delta:Q \times A \rightarrow Q$. We call these states partial because not every attribute may be specified in any state. Thus each partial state subsumes a set of specific states. There is a designated set of states Q_F where the case is finished.

We call T the set of all tasks of the presented case. For each task t there is a set $Q_{0,t}$ of states where this task is reasonable. We express that with the objective function

$$o:T \times Q \rightarrow Q, o(t,q) = \begin{cases} q_F \in Q_{F,t}, & \text{if } q \in Q_{0,t} \\ \emptyset & \text{otherwise} \end{cases}. \quad Q_{F,t} \text{ is the set of all final states for } t.$$

Typically, the Q_0 and Q_F for each t contain only few states. We assume that the learner cannot get stuck, so the following must hold: $\forall q \in Q: \exists t \in T: o(t,q) \neq \emptyset$. With the task *finish* we can define in what states the learner is done and may only leave the case: $Q|q: \forall t \in T \setminus \text{finish}: o(t,q) = \emptyset \wedge o(\text{finish},q) \subset Q_F$.

Additionally we define the availability function $\alpha:Q \rightarrow A$ with $\alpha(q \in Q) = \{a_i \in A \mid a_i \text{ available in state } q\}$. With this function we can express constraints about the UI.

To master a task the learner has to find a list of actions $A_t = (a_1, \dots, a_n)$ so that $\delta(\dots(\delta(q_0 \in Q_0, a_1), a_2), \dots) = q_F \in Q_F$ with $a_{i+1} \in \alpha(q_i)$ and $q_1 = \delta(q_0, a_1), q_2 = \delta(q_1, a_2), \dots$.

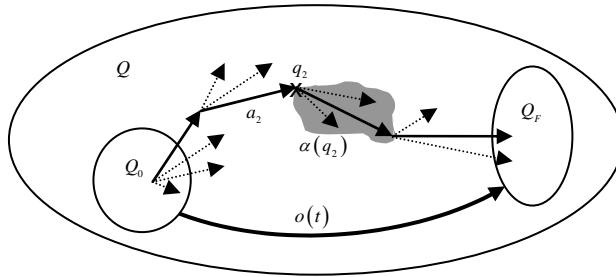


Figure 3: Relations between states, actions and tasks.

As an example, see Fig. 3. To get from one of the states of Q_0 , the learner may take the denoted list of actions (non-dotted arrows) to get to one of the states in Q_F (thus achieving task t). After a_2 , at q_2 she has to choose one of the available action out of $\alpha(q_2)$. We can recognize if the learner acts erroneously if she performs an action that neither does lead to a state in Q_0 of some other task nor stays on a path to the Q_F of the task she currently pursues.

Definitions of the task structure

To get a formal definition of the task structure we define the acyclic graph G_T as the pair (V, E) , where $V = T \cup A$ is the set of vertices, and E is the set of edges between the vertices, given with $E = \left\{ (u, v) : \begin{array}{l} u \in V, \\ v \in A \Rightarrow \neg \exists (x, y) \in E : x = v \\ v \in T \Rightarrow \exists (x, y) \in E : x = v \end{array} \right\}$, so that all leaves in the graph are atomic actions.

Based on this graph we define the structure function

$s : T \rightarrow V, s(t \in T) = \{v \in V \mid \exists (t_i, v) \in E\}$ to calculate the child vertices for a given task.

4 Concepts and overlay model

Based on these definitions we can now formally describe the concepts that the learner has to grasp to master the UI. We discriminate between three different kinds of concepts:

- *structural* concepts: These concepts are about the decomposition of tasks into sub-tasks
- *high level order* concepts: These are the concepts about the objective function of a task. The learner has to find out which tasks are reasonable in a given state.
- *action* concepts: With action concepts we express concepts about the possible A_t s, i.e. sequences of actions to get from a $Q_{0,t}$ to the associated $Q_{F,t}$. These are similar to structural concepts as far as they are about the decomposition of tasks into actions but they also incorporate knowledge about the availability function.

When the learner masters all these concepts, then she knows *when* she may perform a task, *what sub-tasks* she has to do and *how* she can achieve each of them.

The overlay model M that we use keeps track of the learner's competency regarding these concepts with numerical and symbolic scores.

We define M as a 6-tuple $M = (C, S, [n_1, n_2], m_1, m_2, m_3)$, $n_1 \in \mathbb{N}^-$, $n_2 \in \mathbb{N}^+$ where C is the set of concepts, S the set of ordered symbolic values, $m_1: C \rightarrow [n_1, n_2]$ is the numerical score function, $m_2: [n_1, n_2] \rightarrow S$ the symbolic score function and $m_3: S \rightarrow [n_1, n_2]$ is the inverted symbolic function.

To change the overlay model we use rules with complex conditions, whose actions are functions of the form $g: \mathbb{F} \times C \times S \rightarrow \mathbb{F}$, where \mathbb{F} is the universe of all possible functions m_1 and $g(m_1, c, s) = m_1'$ so that $m_1'(c) - m_1(c) = m_3(s)$. We discriminate two types of rules. The *state rules* use predicates to recognize the current partial state(s) or the learner's action sequence, the conditions of *dependency rules* comprise atomic conditions of the form $m_2(m_1(c)) \text{ op } s, c \in C, s \in S$ where $\text{op} \in \{<, \leq, =, \geq, >, \neq\}$.

Example

With this formalism we can specify the set of *state rules* to model correlations like

“If the learner chooses a therapy although she did not rate any diagnoses then the rating for the high level order concept ‘choose treatments for established diagnoses’ is decreased”

and the set of *dependency rules* for correlations like

“If the rating for the action concept ‘diagnoses are added by clicking’ is below some value, then the rating for the action concept ‘therapies are added by clicking’ should also be decreased” (because these two tasks are similar).

5 Assistance by agent interventions

Every time the user performs an action, the overlay model is updated in a two-part step:

- First all state rules are executed. If erroneous or correct behaviour is detected, the values of the associated concepts are changed.
- Afterwards all model rules are executed, but (in contrast to the action rules) these rules may be taken back if their condition is not satisfied anymore and their action may be only executed if they were either never executed or taken back. The cycle of model rules execution is repeated until the model is stable.

After this update step of the overlay model, the didactic components take over. These components decide whether the system should intervene, and if so which of the pre-fabricated interventions should be returned. For these decisions a third set of rules is used.

There is a trade-off between the two main usability challenges concerning this component: On the one hand, the help module should be unobtrusive [J03] so the learner may concentrate on the learning subject. On the other hand, if the learner does something really bad (i.e. something that results in a bad rating for one of the concepts), there should be immediate feedback.

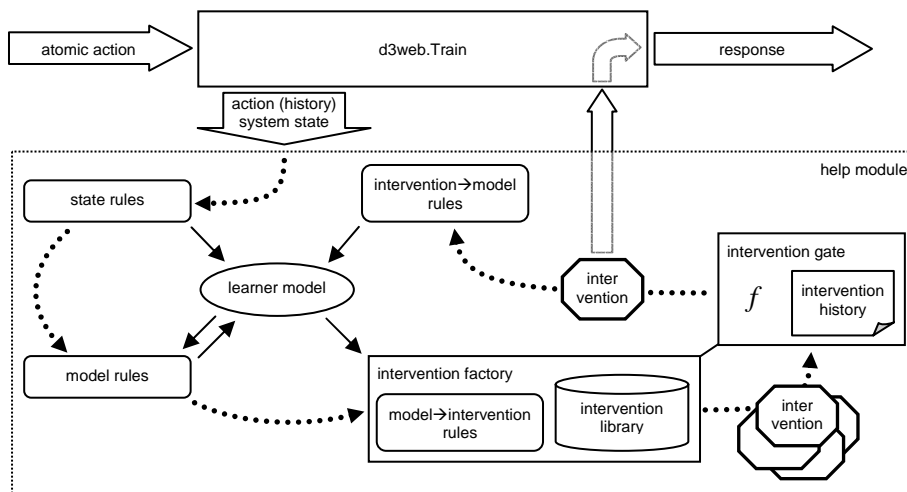


Figure 4: Schematic view of help module

To meet this challenge we use two interweaved didactic components: The *intervention factory* and the *intervention gate* (Fig. 4). The intervention factory uses *model* \rightarrow *intervention* rules, with the same type of conditions like the model rules but with actions that results in the return of a pre-defined intervention out of the intervention library with a specific weight.

The intervention gate thus receives a set of weighed interventions and must decide which of these interventions should pass through. To achieve unobtrusiveness, the intervention gate stores its result in an intervention history – if no intervention passed through a *null*-intervention is stored. Each intervention is stored with its weight, a position number and a timestamp. The intervention gate then computes a score based on:

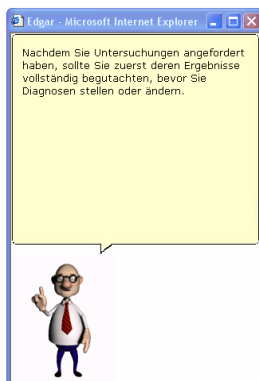
- The summed weight of all identical interventions
- The difference to the number of the last intervention that got through
- The time difference to the last intervention that got through
- The difference to the number of the last identical intervention that got through
- The time difference to the last identical intervention that got through

If this score exceeds a certain threshold, then the intervention passes through, otherwise it is cancelled. If more than one intervention is indicated then the intervention with the highest score is chosen; all others are cancelled by default.

Consequences of interventions

When the training system receives an intervention from the help module the learner's action is cancelled. Instead the tutoring agent appears displaying the content of the intervention which may include links to the (static) help-system.

To model the fact, that a learner probably learns something from the intervention content, now a fourth set of rules comes into play: the *intervention* \rightarrow *model* rules. Since the displayed messages explain one or few concepts it is reasonable to assume that these concept are understood better. Thus we increase the associated values with the same actions used in model rules. Although this may lead to model changes which in turn may trigger *model* \rightarrow *intervention* rules, the latter are ignored at this point.



For example, if the learner ordered examinations which are spread over two sliders in the EPR and immediately starts to diagnose (by doing one of the actions for the diagnose tasks) the help module concludes that he does not know of the high level order concept 'after ordering examinations first view all results then update your working diagnoses notepad' and decreases the associated value. If this value is below some threshold then the appropriate intervention (Fig. 5) may be returned (if it passed the intervention gate) and cancels the learner's action.

Figure 5: The tutoring/assistance agent

6 Discussion and future work

In this paper we presented a solution for the problem, that learners don't have time to learn complex user interfaces - nevertheless interactive simulation based training systems need to have a user interface with a high complexity. We showed how we model the learner's mastery of the concepts behind the interface and intervene appropriately if the learner's actions are erroneous. This way, we allow for learning-by-exploration with a just-in-time assistance guiding the learner. While our system d3web.Train is used in medical domains only, the solution is generic enough for other diagnostic domains too.

The mechanisms described in this paper are partially implemented. While there are complete rule sets for most concepts, the fine-tuning of these rules is not sufficient to allow for an evaluation. Two courses with nearly 400 participants were launched with optimal conditions for an evaluation (200 intermediates and 200 novices with unique accounts). But as the novices in particular may suffer from obtrusive help behaviour the system may only be used for an evaluation if it generates flawlessly unobtrusive behaviour.

Before we started with the UI assistance some of the interventions were hard-coded into the system, e.g. when the learner ordered a set of examinations and did not enter any diagnosis (although there are clues that she should) the action was always cancelled and an appropriate content was displayed by the tutoring agent. Evaluations will hopefully show that the learners prefer a less obtrusive guidance by the system.

We plan to use stereotypes to pre-set the learner model to allow for an initially better behaviour. Additionally we intend to make the learner model transparent to the learner so she can explicitly rate herself.

Acknowledgements

The implementation was mainly done by Reinhard Hatko and Peter Kluegl during a practical course about human-computer interaction.

Bibliography

- [BBH04] Betz, C., Buscher, H.-P., Hörnlein, A., Puppe, F., Schuhmann, M.: Generierung diagnostischer Trainingsfälle aus Arztbriefen, in [P04], 25-36, 2004.
- [CG77] Carr, B., Goldstein, I.P.: Overlays: A theory of modelling for computer aided instruction, Technical Report AI Memo 406, MIT, Cambridge, MA, 1977
- [CY00] Cox, A. L., Young, R. M.: Device-oriented and task oriented exploratory learning of interface designs, in: Proceedings of the Third International Conference on Cognitive Modeling, pp. 70-77. Universal Press, Veenendaal, The Netherlands, 2000
- [d3web.Train] Fallbasierte intelligente Trainingssysteme für die Diagnostik, <http://www.d3webtrain.de> (21.06.2004).
- [FLS85] Fischer, G., Lemke, A., Schwab, S.: Knowledge-based Help Systems, Human Factors, in: Computing Systems – CHI'85 Conference Proceedings, pp. 161-167, ACM, New York, 1985

- [HBP02] Hörnlein, A.; Betz, C.; Puppe, F.: Redesign eines generativen, fallbasierten Trainingssystems für das WWW im d3web.Trainer, in: Bernauer, J.; Fischer, M. R.; Leven, J.; Puppe, F. and Weber, M. (eds.): Rechnergestützte Lehr- und Lernsysteme in der Medizin - Proceedings zum 6. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin, Fachhochschule Ulm, 11.-12. April 2002, ISBN 3-8322-0611-6, Shaker Verlag, Aachen, 2002
- [HRKBP04] Hörnlein, A.; Reimer, S.; Kneitz, C.; Betz, C.; Puppe, F. Semantische Annotierung von Arztbriefen zur Generierung diagnostischer Trainingsfälle, in: accepted in Proc. Of Delfi-2004, "Lecture Notes in Informatics", 2004.
- [J03] Jameson, A.: Adaptive Interfaces and Agents, in: [JS03], pp. 305-330
- [JS03] Jacko, J.A., Sears. A (eds.): The human computer interaction handbook: fundamentals, evolving technologies, and emerging applications, ISBN 0-8058-3838-4, Lawrence Erlbaum Associates, Inc., Mahwah, New Jersey, 2003.
- [Ma03] Martens, Alke: Ein Tutoring Prozess Modell für fallbasierte Intelligente Tutoring Systeme, PhD thesis, University Rostock, Faculty for Engineering, Rostock, 2003
- [Me03] Mehlenbacher, B.: Documentation: Not Yet Implemented, But Coming Soon, in: [JS03], pp. 527-543
- [MS95] Mullet, K., Sano, D.: Designing Visual Interfaces, Communication oriented techniques, ISBN 0-13-303389-9, Prentice Hall, 1995
- [N01] Norman, D.A.: The Design of Everyday Things, ISBN 0-262-64037-6, MIT Press, 2001
- [P04] Pöppel, S., Bernauer, J., Fischer, M., Handels, H., Klar, R., Leven, J., Puppe, F., Spitzer, K. (eds.): Rechnergestützte Lehr- und Lernsysteme in der Medizin: Proc. 8. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin, Shaker, Aachen, 2004
- [R79] Reenskvaug, T.: Models – Views – Controllers, Xerox PARC Technical Note, 1979
- [R91] Rettig, M.: Nobody Reads Documentation, in: Communications of the ACM (CACM) 34 (7), pp. 19-24, 1991
- [R00] Reinhardt, B.: Didaktische Strategien in generierten Trainingssystemen zum diagnostischen Problemlösen, Dissertation an der Universität Würzburg, infix-Verlag, disk 234, 2000.
- [RKT04] Reimer, S., Kneitz, C., Tony, H.-P., Schewe, S., Hörnlein, A., Puppe, F.: d3web.Train: Erste Evaluationsergebnisse zum Einsatz in der Medizinerbildung an der Medizinischen Poliklinik der Universität Würzburg, in: [P04], pp. 155-164, 2004.
- [Y03] Yoshikawa, H.: Modeling Humans in Human-Computer Interaction, in: [JS03], pp. 118-146.