

Intelligent self-adaptation of user interface complexity in a case-based medical training system

Alexander Hörnlein, Frank Puppe
Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik,
Universität Würzburg, Am Hubland, 97074 Würzburg, Germany
{hoernlein, puppe}@informatik.uni-wuerzburg.de

Abstract

Case based training systems are well accepted in medical education [1]. The complexity of the used cases ranges from simple cases, where all symptoms and findings are presented at once and different questions concerning the diagnoses and/or therapies are to be answered, to very complex cases, where the case is presented in many steps, and each step may include several different kinds of tasks the learners have to accomplish: Determining intermediate and final diagnoses, interpretation of multimedia data gathered by examinations, choosing examinations with the presumably best knowledge gain, controlling the therapies in subsequent consultations, while at the same time justifying all actions. Complex cases are more realistic but have the following disadvantages: the user interface (UI) is typically more complex, the duration of execution by the users is prolonged and the amount of authoring work is increased. The trade-off between the advantages of using realistic, complex cases and these disadvantages depends on the preferences, computer literacy and experiences of the user. To be able to match the needs of different users, we present a technique where the UI complexity of a case can be dynamically adapted to the learner's abilities, i.e. a self-adaptable UI.

Key words

Educational Measurement - Problem-Based Learning - Needs Assessment [I02.594]

1 Introduction

The case-based training system d3web.Train [18] is used in blended learning courses for students [3][25][26][27][28][29] and in the context of continuing medical education (CME) for physicians [35] since 2004. For each course or workshop instance an evaluation is conducted, where one of the most important topics of interest is always the usability of the user interface. We found this one of the hardest part in the development of a generic web-based training system, because only for few tasks that the learners have to accomplish common UI metaphors can be applied and inventing new forms of interaction that are immediately understandable even by computer illiterates is difficult [14]. As described in [15] with d3web.Train it is possible to offer cases with different complexity of contents and user interface. This way each learner may find some cases or – if all cases are available in differently complex configurations – for each case the variant with suitable complexity.

Some problems still persist though:

- Novice learners lack the understanding required to choose between different cases or case variants. How may they estimate their competence concerning a UI they have never seen?

- If the system lets them choose and they overestimate their competence, then they would fail due to the too complex UI and have to start a better suited case or the same case with a less complex UI.
- On the other hand, if they choose a case with a complexity so low that it affects the content complexity of the case they may get unchallenged and thus bored of the subject.

So instead of burdening the learner with the (probably impossible) task of choosing the most adequate complexity, we will show how the system can dynamically adapt its complexity so all learners get the most of all cases.

The paper is organized as follows: Section 2 provides details about different types of UI complexity in training systems. Section 3 presents the training system d3web.Train with its ability to have different levels of UI complexity. In section 4 we look at the adaptive assistance system that d3web.Train uses to support learners in cases with complex UI and in section 5 we describe how we use this assistance system to choose if and how the UI should change its complexity. Section 6 closes with a discussion and future work.

2 User interface complexity

We described UI complexity in depth already in [15] so we will only briefly repeat that here.

The procedural complexity of an application can be divided into different kinds: Information complexity, operational complexity and navigation complexity.

2.1 Information complexity

Information complexity deals with the difficulties the user has with acquiring the content of the UI. In applications with huge information load, the information must be structured and fast navigational, so the users can easily identify which part of information is displayed and how they get displayed the information they want. In medical training systems the information consists of the patients' data, background knowledge to support the learners' decisions and didactic annotations. In most systems the data is displayed in a form resembling an electronic patient record (EPR).

2.2 Operational complexity

Operational complexity is about handling the different tasks posed by the system. A case-based medical training system may support the following tasks:

- **Order examinations**, where the learner must choose which examination is reasonable in the current situation. The observance of cost, time, patient risk, etc. may be requested.
- **Interpret examinations' results**. Examination results may not immediately be available. Instead the intermediate results like radiographs may be presented and the learners have to find the definite result themselves.
- **Diagnose**. Based on the examination results learners have to choose working or final diagnoses, possibly with different annotations like "*possible*", "*probable*", "*improbable*" and different states like "*decreasing*", "*increasing*", "*stable*".

- **Choose treatment.** Based on diagnoses and examinations' results the correct treatment must be found and possibly categorized. While the output of the diagnose task is a single diagnosis or a list of diagnoses, the treatment must cope with alternatives like "*EITHER drug A AND drug B OR drug C AND TWO OF drug D, drug E, drug F*". On a lower level of detail dosing, allergy check, etc. becomes important.
- **Monitoring.** While this is no task like the aforementioned, it is important to discriminate between the first diagnose and treatment steps and the following monitoring steps, because typically there is no full examination step needed, mostly the same diagnoses must be newly rated and the treatment must be adapted to the new patient state.
- **Justifying input.** The easier acquirable part of learning contents is the knowledge of the correct solutions for different tasks, the harder is having an understanding why these solutions are correct. To control these learning targets the learners have to give explanations for their solutions. Explanations may be given in a variety of forms, beginning with answers to multiple choice questions and up to concept maps like in [16].
- **Tasks about background knowledge.** These tasks are used to guide the learners through the case by preparing for following difficult tasks or to control the reaching of learning targets at the end of the case. Just as the justifying input task the variety of forms is unlimited.

There are many different task and many different ways to present a task. Many of these presentations use ways of input that differ from the ones the users are accustomed to, so the complexity of the UI increases with each unknown form of interaction the users face.

2.3 Navigational complexity

Apart from the complexity to handle the tasks it can be difficult to get to the different tasks or to revisit a task to check or correct the given solutions or reread the system's feedback. There are many different ways imaginable how the user may be able to navigate the tasks so we will present only some alternatives from the spectrum of possibilities:

Realistic navigation: When the system has the claim of being a simulation of a physician's work, then there will not be any central instance showing the tasks' states. Instead the tasks appear at locations appropriate to situations when the real physician has to face them. For example, if there are results to interpret, then there will be a link or a button next to the results with "*please interpret these results*".

Linearized tasks: In page-based systems (like Casus [17]) there is only minimal task navigation. Each page of such systems poses a number of tasks in a given order. Since the user never has to handle multiple tasks simultaneously, there is no need to navigate between different "open" tasks. The navigation to a previously processed task is by navigating back to the page with the wanted task.

Central task list: Every time a task is posed it is added to a central task list where each entry links to the corresponding task. The task list may also show the states of the tasks, may be ordered by task types or by the time when the task was added or similar.

Central navigation: Since the task list may become overcrowded with entries, and thus take too much precious screen space, a more structured approach may be necessary. First, the possibility to revisit any task does not need to be available at any time, since e.g. questions

about background knowledge are of no immediate importance after this knowledge came to use. Or if the case consists of many diagnostic/therapeutic cycles the tasks of a previous cycle are rather unimportant for the current ones so the navigation to those tasks can be more tedious than navigation to one of the current tasks.

Each of these examples has its advantages and disadvantages, the decision is always a trade-off between realism, screen real estate and simplicity. The simpler alternatives have also an impact on the didactics: When the learner can not choose between tasks he may not learn how to choose.

3 The training system d3web.Train

As stated in the introduction, d3web.Train is a case-based training system for general diagnostic/therapeutic problems with some special tailoring for medical domains. Due to its history d3web.Train supports cases with and without backing formal expert knowledge: d3web.Train started as reimplementation of D3Trainer [18][19], a training add-on for the expert system shell D3 [20], which relied heavily on knowledge (mostly in the form of heuristic rules) to generate dynamic tasks and dynamic feedback to the learners' choices. As d3web.Train got used more and more from lecturers not familiar with D3 and its way to enter highly formalized knowledge and problems concerning the reuse of big knowledge bases emerged [21] the pressure increased to allow for cases with less formal and more informal knowledge. In a recent dissertation [24] a way to generate training cases from specially revised medical discharge letters was created – the resulting cases contain barely structured content only and thus not enough knowledge to allow for a quasi-simulative environment (like D3Trainer does). Although they contain less knowledge, these cases match the lecturers' and students' needs especially well: As most of the students are in their early clinical semesters they lack the knowledge to master some of the tasks the knowledge based cases pose (e.g. deciding which examination is optimal in a certain context concerning differential diagnosis, risk, time, costs, etc.). Additionally they would not get much gain from the generated feedback since the explanations from the expert system do not match the structure of their knowledge so feedback especially prefabricated for the present case is preferable.

3.1 Technical background

d3web.Train is web-based (and thus, server-based) which has the advantage that users do not need any installations or updates and thus can use the training system almost anywhere where they have access to the supported browser (Microsoft™ Internet Explorer™ 5.5+). On the other hand the web-basedness poses some problems when trying to apply cutting edge UI techniques (like 3D-navigation, transparent layers, etc.). Because with the constraint that data from the server is only released when the client really needs it (to prevent cheating) all implementations are prevented where complete data is initially sent and is hidden from and revealed to the user by Javascript [7]. So there is a decreased responsiveness dependent on the clients' bandwidth. The implementation counters this with an elaborate framework where only changed or new page content has to be served and unchanging content once served is cached on the client-side for reuse. Additionally, d3web.Train makes heavy use of Javascript to allow for as much types of information, navigation and tasks as reasonable (and wanted by lecturers). Since d3web.Train is highly configurable and provides interfaces for plug-ins or replacements of existing modules (e.g. user management, case management) the training system can easily be adapted to suit very differing needs.

3.2 Types and forms of navigation and tasks in d3web.Train

For an overview of all different abilities of d3web.Train one has to discriminate between cases with and without a comprehensive knowledge base, i.e. knowledge based cases with highly structured knowledge and cases based on terminology only with less structured knowledge:

ability	knowledge based with highly structured knowledge	terminology based with less structured knowledge
types and forms of navigation		
realistic linking by scattered buttons/links	✓	✓
central navigation by task buttons (disabled if task is not currently present, hidden if task is never used in the current case), realistic linking as shortcuts, enhanced with task lists	✓	✓
minimal navigation (“ <i>next</i> ”)	✓ seldom makes sense	✓
types of tasks		
order examination		
by selecting single examinations	✓	✓
by ordering next group of examinations	✓	✓
by ordering next group of examinations and selecting single examinations in the last case section	✓	✓
interpret	✓	✓
justify interpretation	✓	✓
choose diagnoses	✓	✓
justify diagnoses	✓	✓
set therapies	✓	✓
answer questions about background knowledge at end of section		✓
answer questions about background knowledge after case conclusion		✓
answer questions about background knowledge anytime		Planned
adapt diagnoses in follow-up sessions		✓
adapt therapies in follow-up sessions		✓
forms of tasks		
choose terminology elements (diagnoses, therapies, examinations)		
from hierarchical long menu (HLM) (see 5.1.1)	✓	✓
from long menu (LM) (see 5.1.1)	✓	✓
as text input	✓	✓
as generated multiple/one choice (MC/OC) questions		Planned
as authored MC/OC question		Planned
with String search	✓	✓
with synonyms search	✓	✓
with categorization		
of diagnoses (“ <i>suspected</i> ” ↔ “ <i>established</i> ”)	✓	✓
of therapies (“ <i>reasonable</i> ” ↔ “ <i>indicated</i> ”)	✓	✓
result interpretation with generated MC/OC/number questions	✓	
result interpretation with authored MC/OC questions		✓
background knowledge tasks as MC/OC questions		✓
diagnoses/therapies adaptation in follow-up sessions		✓ like “choose terminology elements”
justification of interpretations by marking regions in images for each answer	✓	
justification of diagnoses by marking important text passages		✓
with categorization (“ <i>strong evidence</i> ”, “ <i>evidence</i> ”, “ <i>evidence against</i> ”)		✓

task feedback		
present correct solution and rate user input (all supported tasks)	✓	✓
prefabricated explanatory feedback (containing multimedia elements)		
for result interpretation with authored MC/OC questions		✓
background knowledge tasks as MC/OC questions		✓
generated text explanations for choosing terminology elements	✓	

Table 1: Overview of d3web.Train’s abilities as regards navigational and operational diversity. On the right hand it is marked which ability is supported by the two types of cases.

4 The adaptive assistance system

To support users facing UI variants with higher complexity, d3web.Train has an adaptive help system available in the form of an intelligent avatar displaying helpful information when it recognizes user errors. This system is described in [4] and evaluation results are available in [5], but a short summary is needed: We implemented a help system that monitors the users’ actions and the system state and builds a user model with overlay modelling [22]. Based on this model it decides if and what help is needed. When the user triggers an action that would lead to a fatal situation (like going to the next case section without handling all tasks of the present section) the action may be even aborted. The help system is highly configurable to allow stricter modes where each erroneous action is blocked and help is displayed or looser modes where the user is allowed to make errors and only seldom is interrupted.

5 Intelligent complexity adaptation

With the ability to present information, tasks and navigation with different forms and complexity and the ability to detect if the user has problems with a certain UI concept we may now go one step further: Instead of displaying help the help system will now be used to determine if the UI complexity should be decreased or may be increased. To be able to start the first case with suited complexity each user will have to fill out an initial questionnaire about prior experiences with the system and overall computer literacy.

While the **informational complexity** will not be subject to self-adaptation there may be ways to adapt it quantitatively: In cases with highly structured knowledge there is a notion of abnormality of examination results, so the system can be configured to only display the abnormal, more abnormal or most abnormal results. As the patient’s data thus gets fleshed out, the user can make his decisions based on pathological results only.

When adapting the **operational complexity** the system will change the forms of tasks as suggested by user errors, the **navigational complexity** will switch between realistic navigation, central navigation and minimal navigation. As user errors point to less complex alternatives only, the system tentatively increases the complexity if no user errors happen for a certain time with the consequence that in the least complex mode (where no errors are possible) the next more complex alternative will always be chosen after this time.

But as there are reasons why not any complexity decrease is possible without added knowledge, too, in the following sections we will not only describe what errors the user can make and which implications on adaptation they have, but will also give the knowledge constraints for transformations of tasks and navigation.

5.1 Adaptation of operational complexity

Some task types can be grouped together based on their similarity, namely all tasks dealing with choosing elements from one of the terminology of diagnoses, therapies or examinations, and all tasks that are available in MC/OC question form only. Two tasks, justifying result interpretations and justifying diagnoses stand out and must be handled separately.

5.1.1 Choosing terminology elements

As can be seen in table 1 these tasks are the ones where the UI is most flexible: Already available forms of these tasks are as HLM, LM or text input, planned are MC/OC questions. The hierarchical long menu (HLM) form is a Microsoft Windows Explorer-like structure while the long menu (LM) [6] form is a very long flat list of alternatives. With short menu (SM) we denote a LM variant with a shorter list of alternatives. There are additions to help the user like String search or synonyms search. To allow for finer solutions a further categorization can also be requested.

In UI complexity order where HLM is the most complex form and text input the simplest:

HLM → LM → MC → OC → text input

5.1.1.1 Transformation between HLM, LM, SM, MC, OC and text input

Transformation from **HLM to LM** is straightforward, as all terminology elements must be put in one flat structure and typically ordered alphabetically. To generate a HLM form without hierarchical knowledge is not possible.

For an intermediate form between MC and LM, a **SM** has substantially more alternatives than an MC/OC question but less than a proper LM question, so instead of just adding all HLM elements we may have to discard some, which is possible if the correct solutions are no leaves in the hierarchy tree. Then we can discard all elements of higher depth.

The transformation to **text input** can be straightforward: The system only has to match the user input with each alternative. But the resulting assessment would not be really fair: The use of a synonym, a writing variant (“*Koloskopie*” vs. “*Coloskopie*”) or a spelling mistake would decrease the users’ score more than didactically reasonable. So if a certain level of feedback quality has to be achieved, further (domain specific) knowledge to enable techniques of natural language processing (NLP) is essential.

When there is a text input form with such knowledge then the LM form is created by using the complete terminology as the list of alternatives.

Transformations to MC question

A very easy form of answering questions is as multiple/one choice questions since this is the form most often faced in daily life: There are few alternatives from which the user has to choose one or more by marking them. The didactic disadvantages of MC/OC questions are discussed in [6]. The problem when transforming from text input or LM form (with several hundred supported entries) to a question with only about 5-10 alternatives is the problem of choice: Which wrong alternatives can be paired with the correct solutions so that the correct answers can not be singled out too easily?

Without hierarchical knowledge there is basically only guessing. By using naïve techniques like fuzzy equality we could generate alternatives with similar writings but the semantic similarity would be dubious (“*Hypothyroidism*” vs. “*Hyperthyroidism*”) so there is no intelligent rating possible with the possible consequence that alternatives chosen by the user have nearly the same meaning as the correct solution but may be rated as 0% correct.

With more sophisticated NLP knowledge the system may find related alternatives to the correct solutions (like “*Acute transmural myocardial infarction of inferior wall*” as a partially correct alternative to “*Acute transmural myocardial infarction of anterior wall*”), and this knowledge may then be used to find a more reasonable rating.

Since d3web.Train uses a closed world assumption (i.e. the negation of a diagnosis or therapy is implicitly done by not mentioning the diagnosis or therapy at all) with HLM knowledge only we can create and rate partially wrong alternatives, as the distance in the hierarchy can be used to find adequate alternatives. The closed world assumption is necessary to be sure that a sibling of a correct solution (with a distance of 2) is not its preclusion.

When there is no HLM or NLP knowledge available then an approach different from transformation is possible: The authoring of the alternative task form, where the lecturer manually chooses alternatives and rates them. d3web.Train supports the MC/OC question form for some tasks so the support of the MC/OC form as a variant for other tasks can easily be added. This requires increased authoring cost, but on the other hand no other knowledge (that must be acquired, too) is needed and the resulting MC/OC question may be much better – at least didactically.

Transformation to OC question

One choice questions are implemented as common HTML radio form elements, where only one answer can be selected. If there is more than one correct answer, then there must be some knowledge available to differentiate between the correct answers. Therefore, d3web.Train has a notion of weight, which may be used to mark one answer as more important than the others. This answer is then used as the one correct answer. The rating module can observe the different weights for a more balanced rating. It is important to explain to the user that he has to choose the currently most important terminology element.

5.1.1.2 Criteria for self-adaptation

The certain detection for **HLM handling problems** is based on additional information presented to the user and the analysis of his actions: We differentiate between three granularity levels for terminology elements

- fine: all leaves in the hierarchy tree
- rough: direct children of the (unavailable) hierarchy root
- intermediary: all elements neither fine nor rough

and the number of expected solutions from each level is displayed as a hint. If an opening of hierarchy nodes is required to enter the displayed number of solutions and this opening does not take place then the help system assumes that the user does very likely not understand the HLM concept so the system should switch to a form where the alternatives are easier to select.

The handling of **LM questions** is not very error-prone: One has to select multiple entries from a long list and the task is done. In case of d3web.Train the user has to navigate to the desired entry and click on “add” to add the entry to the list of his solutions. When the list is complete the user has to click on “submit” and the task is finished. In the LM form there is

a hint displayed, too: The number of expected solutions (in each category if there are categories). Still the user may make the error of submitting too few or too many solutions. But it is hard to pinpoint a deficit in UI handling this way – it is more likely that the error happened due to domain knowledge shortcomings.

But there may be **another reason** why the UI should pose the task in another form: Proper LM questions must contain hundreds of alternatives. So the use of LM questions is only reasonable if the case author can be sure that the learners know and use the correct terminology or the list of alternatives must contain many synonyms and maybe even variants in writing. Otherwise the user may not find the entries he searches for and may not be able to finish the task correctly. The same holds for the simplest text input form which may be more robust against spelling mistakes but in effect must also have such a list, whether generated or explicitly authored. When this list is not present, the system must use the MC/OC form, as it is less complex than the HLM form.

Since there is a hint on **MC questions** how many answers are expected, the user may erroneously override this and add the wrong number of items. This can be prevented with OC questions where the choice of more than one item is technically impossible.

5.1.2 Other tasks

As can be seen by the length of section 5.1.1 a detailed explanation of the simplification of the other tasks would go beyond the scope of this work, so we give only a very brief synopsis:

Justification of interpretations: In the default most complex form, the user has to find images with evidence for the user's answer in the result interpretation task and has to mark the interesting regions by drawing multiple rectangles. Easier forms are pre-generated images based on the author's markings and additional fake marking which must be chosen like in the MC/OC form or single-answer forms where only the single most important marking has to be entered by the user.

Justification of diagnoses: In the most complex form the user has to find important text passages in the EPR and has to add them to a list of evidences for his diagnoses. Easier forms are MC/OC questions generated from all text passages and fake passages added by the author. These questions can request evidential passages for each diagnosis or all diagnoses where a passage is evidential for. A single-answer form where only the most important text passage \Leftrightarrow diagnosis relation must be entered is also reasonable.

Other tasks with MC/OC form: All considerations made for the MC/OC question form in the terminology element choosing task can be applied here, with the limitation that hierarchical knowledge is not available.

5.2 Adaptation of navigational complexity

As shown in table 1 there are three forms of navigation (ordered by complexity) supported in d3web.Train: realistic by links scattered in the EPR and on other elements of the UI, explicit by a central navigation hub with task lists or minimal where the user can not choose which task he wants to handle next.

Transformation to central navigation hub

To allow for a central navigation hub with the features it has in d3web.Train – displaying only tasks that are used somewhere in the case and (visually and technically) disabling those that

are not currently available – there is not much preparation required. The case content must be analysed at some point before the case is started to see which tasks are used. The main difference between the realistic navigation and the central navigation however needs more work: In realistic navigation the links go to single tasks only while in central navigation they often go to groups of tasks. So the central navigation hub consists not only of the navigation part to the different types of tasks but also of lists of tasks reachable by the central navigation.

Transformation to minimal form

The further elimination of navigation to almost no navigation (apart from “next task”) needs special considerations about the ordering of these tasks – the computation which tasks are to be posed in each case section already has to take place for the transformation step to the central navigation hub form. While the navigation buttons in central hub are in an order following loosely the select and test model (ST model, see below) [12][11], the user is not forced to process the tasks in this order. So the ordering of the central navigation must not be adapted per case (section).

In contrary, in the minimal form a reasonable ordering of task is essential. The ST model is a generic plan for patient care where all steps like diagnosis, treatment, etc. are put together in the form of connected cycles, where the start of each cycle is triggered by new examination results.

Since the minimal navigation form only makes sense in cases where the user can not freely order examinations but instead gets sets of examination results at the start of each section, the task order of each section follows one cycle in the ST model:

- Data Observed/Expected Data & Request New Information
(≈ Order examinations)
- (Abstraction to) Clinical Evidences
(≈ Result interpretation)
- Diagnostic Hypothesis & Structuring Diagnostic Space
(≈ Choose and categorize diagnoses)
- Therapies & Ranking
(≈ Choose and categorize therapies)

The other tasks supported by d3web.Train are special to training systems and are added appropriately – with the exception of background knowledge tasks which are always set to the end of a section. As of now this poses some problems to the case authors, but with the support for arbitrarily positioned background knowledge tasks we will give the author the possibility to accompany each task with background knowledge tasks.

Detection of navigation errors

Two kinds of errors concerning the navigation are possible: The user misses some tasks or he finished the task in the wrong order (like choosing diagnoses although not all examinations results are interpreted). The first one is easily detectable and suggests simplification to explicit navigation and – if the errors occur afterwards – minimal navigation. Errors of the second kind are also detectable but the implications are not so clear: The order of tasks is on first sight not part of the UI handling knowledge but belongs to the medical domain, so it is arguable if the system should be adapted based on errors concerning the case content. On the other hand d3web.Train supports the ST model, which is applicable to almost any medical domain or diagnostic problem solving process. So as this process is part of d3web.Train’s structure, the knowledge about it is in fact UI handling knowledge. Thus, if the user handles

the tasks in a different order than proposed by the system, then we assume that he does not know about the order and may switch also to explicit navigation (where the order of tasks is displayed) and onward to minimal navigation.

6 Discussion and Future work

We presented a concept how the training system d3web.Train can adapt itself to accommodate users with different knowledge about handling d3web.Train's UI and discussed the implications and constraints of such adaptations.

As previously conducted evaluations have shown it is hard to find a course configuration where all users are content with. We hope that a self-adapting system will accomplish this.

When we consider the knowledge constraints for the specified transformations it becomes clear that a self-adapting system can only be realised with enhanced case content, thus some more work by the case authors is required – but we are confident that the authors will take the effort as such enhanced cases will make special cases for novices obsolete.

Of course, some implementation is needed, too:

- We need to integrate NLP modules to handle tasks in text input form well. Several diploma theses addressed related issues [30][31][32][33][34]
- We have to enable the author to add background knowledge task at any position at the progression of the case
- There must be the possibility to manually add MC/OC questions as (redundant) alternatives to other task forms
- We must find good heuristics to generate the MC/OC question form from other forms

We are planning to evaluate a prototypical system in the near future to find out if a self-adapting system gets accepted or if the system changes will irritate the users. The outcome – that possibly not all of the discussed transformations equally improve the user satisfaction and that some of them create more irritation than satisfaction – will then be used to create a revised system with optimal acceptance and user satisfaction.

We hope to be then one step closer to a training system with rich and complex cases which can be used anywhere, anytime instantly from any user.

Bibliography

- [1] Mathies H, Fischer M, Haag M, Klar R, Puppe F.: eLearning in der Medizin und Zahnmedizin, Proc. 9. Workshop gmds AG CBT in der Medizin, Berlin: Quintessenz, 2005
- [2] Betz, C.; Hörnlein, A.; Puppe, F. Experiences with generating diagnostic training cases from dismissal reports, in [1], p. 50-58, 2005
- [3] Reimer, S.; Hörnlein, A.; Tony, H.-P.; Krämer, D.; Oberück, S.; Betz, C.; Puppe, F.; Kneitz, C. Assessment of a case-based training system (d3web.Train) in rheumatology "Rheumatology International", Springer Verlag, Apr 2006: 1-7, DOI 10.1007/s00296-006-0111-x, URL <http://dx.doi.org/10.1007/s00296-006-0111-x>, 2006
- [4] Hörnlein, A.; Puppe, F. Applying learner modelling for user interface assistance in simulative training systems, Proceedings of the workshop "Teaching and Learning Systems: The Role of AI" at the 27th German Conference on Artificial Intelligence (KI2004) in Ulm, 2004
- [5] Hörnlein, A.; Puppe, F. Evaluation eines adaptiven Hilfesystems für Bedienprobleme mit d3web.Train in H. K. Matthies, M.R. Fischer, M. Haag, R. Klar, F. Puppe (eds): eLearning in der Medizin und Zahnmedizin, Proceedings zum 9. Workshop der GMDs AG Computergestützte Lehr- und Lernsysteme in der Medizin, Freiburg, 13. September 2005, ISBN 3-87652-899-2, Quintessenz Verlags-GmbH, Berlin, 2005

- [6] Schuwirth, L. W. T., van der Vleuten, C. P. M., Stoffers, H. E. J. H., and Peperkamp, A. G. W. Computerized long-menu questions as an alternative to open-ended questions in computerized assessment. *Medical Education* 30, 50-55, 1996
- [7] Standard ECMA-262, ECMAScript Language Specification, Website, <http://www.ecma-international.org/publications/standards/Ecma-262.htm>, 2007, [as available on 2007-02-23]
- [11] Ramoni, M., Stefanelli, M., Magnani, L., Barosi, G., An epistemological framework for medical knowledge-based systems, *IEEE transactions on systems, man, and cybernetics* (IEEE trans. syst. man cybern.), 1992, vol. 22, no6, pp. 1361-1375 (99 ref.), Institute of Electrical and Electronics Engineers, New York, NY, 1992
- [12] Kindler, H., Densow, D., Fliedner, T.M., A pragmatic implementation of medical temporal reasoning for clinical medicine, *Computers in Biology and Medicine* 18 (1998) 105-120, Elsevier Science Ltd, 1998
- [14] Norman, D.A., *The Design of Everyday Things*, The MIT Press, 1998
- [15] Hörnlein, A., Puppe, F., Reduzierung der Komplexität der Benutzeroberfläche von Trainingssystemen, *GMS Medizinische Informatik, Biometrie und Epidemiologie*, 2006; 2(3):Doc19, 2006
- [16] Brüchner, K., Schanze, S., Holznecht, C., Einsatz und Evaluation eines computerbasierten Concept Mapping Templates in der medizinischen Lehre, in: Pöpl, S., Bernauer, J., Fischer, M., Handels, H., Klar, R., Leven, J., Puppe, F., Spitzer, K. (eds), *Rechnergestützte Lehr- und Lernsysteme in der Medizin*, Proceedings zum 8. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin, Universität zu Lübeck, 25.-26. März 2004, Shaker Verlag, 2004
- [17] Fischer M., Schauer, S., Grasel, C., Bähring, T., Mandl, H., Gartner R., Scherbaum, W., Scriba, P.C., Modellversuch CASUS, ein computerstütztes Autorensystem für die problemorientierte Lehre in der Medizin, *Zeitschrift für ärztliche Fortbildung (Z. ärztl. Fortbild.)*, 1996, vol. 90, nr. 5, pp. 385-389, 1996
- [18] 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
- [19] Reinhardt, B., Didaktische Strategien in generierten Trainingssystemen zum diagnostischen Problemlösen, Dissertation an der Universität Würzburg, infix-Verlag, disk 234, 2000
- [20] Puppe, F., Gappa, U., Poeck, K., Bamberger, S., *Wissensbasierte Diagnose- und Informationssysteme*, Springer, 1996
- [21] Baumeister, J., *Agile Development of Diagnostic Knowledge Systems*, DISKI 284, IOS Press, October 2004, 272 pp, ISBN: 978-1-58603-463-4, 2004
- [22] Carr, B., Goldstein, I.P., *Overlays: A theory of modelling for computer aided instruction*, Technical Report AI Memo 406, MIT, Cambridge, MA, 1977
- [23] Rettig, M., Nobody Reads Documentation, in: *Communications of the ACM (CACM)* 34 (7), pp. 19-24, 1991
- [24] Betz, C., Scalable authoring of diagnostic case based training systems, Dissertation an der Universität Würzburg, 2006
- [25] Hörnlein, A., Puppe, F., Schnabel, C., Völker, W., Vergleich komplexer und einfacher Trainingsfälle in der Kardiologie, in M. Löffler, A. Winter (eds): *Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie e.V., 51. Jahrestagung*, 10.-14. September 2006, Jütte-Messdruck Leipzig GmbH, Leipzig, 2006
- [26] Krämer, D., Reimer, S., Hörnlein, A., Betz, C., Puppe, F., Kneitz, C., Evaluation of a novel case-based Training Program (d3web.Train) in Hematology, "Annals of Hematology", Springer Verlag, 2005
- [27] Reimer S. Hörnlein, A., Tony, H.-P., Kraemer, D., Betz, C., Puppe, F., Kneitz, C., Evaluation of a Case-Based Training System (d3web.Train) in Rheumatology, in H. K. Matthies, M.R. Fischer, M. Haag, R. Klar, F. Puppe (eds): *eLearning in der Medizin und Zahnmedizin*, Proceedings zum 9. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin, Freiburg, 13. September 2005, Quintessenz Verlags-GmbH, Berlin, 2005
- [28] Reimer, S., Puppe, F., Hörnlein, A., Tony, H.-P., Kneitz, Ch., Implementierung eines fall- und webbasierten Trainingssystems in das Curriculum der klinischen Immunologie/Rheumatologie, Poster session, 111. Internistenkongress, Wiesbaden, 02.-06.04.2005, Deutsche Gesellschaft für Innere Medizin e.V., 2005
- [29] Reimer, S., Kneitz, C., Tony, H.-P., Schewe, S., Hörnlein, A., Puppe, F., d3web.Train: Erste Evaluationsergebnisse zum Einsatz in der Mediziner Ausbildung an der Medizinischen Poliklinik der Universität Würzburg, in S. Pöpl, J. Bernauer, M. Fischer, H. Handels, R. Klar, J. Leven, F. Puppe, K. Spitzer (eds): *Rechnergestützte Lehr- und Lernsysteme in der Medizin - Proceedings zum 8. Workshop der GMDS AG Computergestützte Lehr- und Lernsysteme in der Medizin*, Universität zu Lübeck, 25.-26. März 2004, Shaker Verlag, Aachen, 2004
- [30] von Schoen, P., Automatische Aufbereitung von Arztbriefen für Trainingsfälle mittels Anonymisierung und Terminologie-Matching, Diplomarbeit, Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik, Universität Würzburg, 2006
- [31] Braun, C., Informationsextraktion zur erwartungsgesteuerten Diagnosecodierung, Diplomarbeit, Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik, Universität Würzburg, 2006
- [32] Omasreiter, M., Entwicklung und Beurteilung verschiedener Systeme zur Informationsextraktion aus semi-strukturierten Dokumenten, Diplomarbeit, Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik, Universität Würzburg, 2004
- [33] Bold, B., Natürlichsprachliche Datenerfassung in komplexen Dialogsystemen: Integration und Evaluation, Diplomarbeit, Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik, Universität Würzburg, 2006
- [34] Dressler, A., Diagnosekodierung medizinischer Freitexte – am Beispiel sonographischer Befundberichte, Diplomarbeit, Lehrstuhl für Künstliche Intelligenz und Angewandte Informatik, Universität Würzburg, 2005
- [35] CME-Rheumatologie, Zertifizierte medizinische Fortbildung Online, Website, <http://www.cme-rheumatologie.de>, 2007 [as available on 2007-02-27]