Interaction Design and Architecture(s) Journal - IxD&A, N. 9-10, 2010, pp. 22-36

Plug's Train – A component Based Approach

Alke Martens, Dennis Maciuszek, Geraldine Ruddeck, Martina Weicht

University of Rostock, Dep. of Computer Science and Electrical Engineering,
Albert-Einstein-Str. 21, 18059 Rostock, Germany
alke.martens@uni-rostock.de

Abstract. Technology enhanced learning can look back on a comparably long tradition. Surprisingly, the software systems in this field are seldom constructed based on state-of-the art insights in software engineering. Using basic software engineering techniques involves system descriptions on a formal and abstract level, to use patterns for software development, and to implement the system in a re-usable and adaptable way. Our component based framework JaBInT (Java Based Intelligent Tutoring), which is based on a firm software engineering ground, allows for re-usability, adaptability, and flexibility in system development -- and might thus be a basis for teaching and training software used in ubiquitous learning as well.

Keywords: component based design, framework, game-based learning

1 Introduction

Several years ago, in the context of the research project Docs 'n Drugs -- The Virtual Hospital [21] we found that developing an Intelligent Tutoring System (ITS) in our case meant to partly re-invent the wheel. We investigated several ITSs and also several theoretical descriptions of the ITS architecture (e.g. [1], [4], [15]). The result has been the insight that the ITS core architecture is more or less fixed. The so-called classical ITS architecture [23] has been described by Clancey in the 1980s [5] and, with only small changes, is also valid today. This basic architecture consists of four models, i.e. learner model, expert knowledge model, pedagogical (knowledge) model, and user interface. We looked deeper into ITS development and focused on the role and functionality of each of the models which constitute an ITS. The result of this has been a surprise: Whereas the basic role seems to be fixed based on Clancey's description [5], the realizations vary. The variations are not only due to technical demands (e.g. pedagogical agents [14], simulations [12], [14], [26]) or based on domain specific aspects (e.g. grammar [25] vs. medicine [3]), but they also seem to have an evolutionary aspect. Even if the main architecture is based on Clancey's description, the role and functionality of each of the modules is flexibly adapted to what is currently needed. A very good example for this can be seen in the expert knowledge model. The expert model started historically for example with MYCIN -- the medical expert system, which has been used as the basis of GUIDON [6]. The first expert knowledge model was a sketch of the knowledge of a physician. In the beginning, GUIDON has been restricted to medical diagnosis finding. The expert knowledge

model thus contained facts and rules related to medical diagnoses. From the software engineering perspective, the expert knowledge model in GUIDON has been realized as an expert system. It has been implemented as own partial systems in the ITS, i.e. it is more than a mere database. Part of the system's "intelligence" is located here. Other applications, e.g. [14] decided to "outsource" the "intelligence". Their expert knowledge model still contains declarative and procedural knowledge about the application domain, but is no longer able to act on its own. It is only a (comparably complex) database, which is accessed by another system component. The same happens with the pedagogical knowledge model. Regarding the number of all investigated systems (e.g. in [23] and in [27]), in most cases we found a more or less simple extension of the expert knowledge model, sometimes pedagogical agents with own interface, steering component and database are embedded and -- in the sum -- provide the pedagogical knowledge. Especially in the latter case we wondered, whether or not such an agent can be separated from its original system and re-used (without deep modifications) in another system.

From the perspective of our research in the late 1990s, this was surprising, as one idea about developing different sorts of ITSs for the same application domain should (scientifically seen) be to enhance the system's functionality, and not just to re-implement existing ideas. But how could existing approaches be evaluated, if their core functions are not explicated? Also, this was the time of upcoming discussions about software engineering, object oriented design and the target goal of all complex applications: re-usability and structures, which are easy to communicate. Both claims of state-of the art software cannot be met with most of the ITSs of the last years!

We extended our research towards other eLearning systems (those who are not said to be "intelligent"). We found that also these systems have the same drawback: whereas basic system parts are more or less the same on the abstract level, the realizations vary, most of the time with similar underlying ideas, in most cases not carefully documented and in almost all cases not re-usable (neither software nor ideas). Regarding the fact that one of the most important things to do in the future of technology enhanced learning (TEL) is to empirically evaluate the effects of learning with modern technology, this will be a big problem: how can one compare the effects of using different systems, if the main system design is not comparable at all?

We started to look closer at different fields of TEL and found, that the main components which we have located in our ITS research can also be retrieved in the "typical" eLearning system [22]. We have excluded learning management systems from these investigations, as they have another purpose and another structure.

Then the topic of ubiquitous and pervasive learning arose -- and liquid learning, focusing on liquid learning places, mobility, adaptability and the like. Looking at our topic from this perspective, we came up with the same insight: will there be again a pure evolution of specialized domain dependent systems, which cannot be compared, and neither re-used, nor adapted? From our perspective, this would violate the main idea of liquid learning. Thus, we will describe our component based approach as a means to develop future systems which are at their core functionality domain independent, which can be extended and re-used, and which are described and designed in a way that allows for scientific comparison and empirical evaluation. Means we have used and developed, but which we cannot describe in detail in this article, are patterns for ITSs and other teaching and training systems (see references

[8], [9], [11], [13], [19] and [30]), a formal Tutoring Process Model for describing the adaptation process for adapting content and navigation to a learner (see references [23] and [24]), and the connection to the modeling and simulation component based framework JAMES II [10].
In the following, we will describe the JaBInT (Java Based Intelligent Tutoring) approach, the plug 'n train concept, and we show the viability of our approach by "proof of concept" implementations. The applicability of our approach to the game-based domain will be shown. Finally, we will discuss the usage of JaBInT for ubiquitous learning.

2 Basic Architecture

In this text we use four major terms: model, framework, component and module. The *model* is an abstract description of something, for example the formal tutoring process model, the architecture model or the component model. The *components* are software components [31]. Software components are part of a composition (a software system or a framwork) and can be independently deployed by other software developers. Components are developed based on some sort of component model, which describes their basic structure, the kind of interfaces and the communication. A well-known example for a component based approach is the Eclipse framework (see www.eclipse.org). Similar to Eclipse, we have developed our system as a *framework* (e.g. in contrast to a service-oriented architecture). In software engineering, the term framework is used to describe a set of re-usable structures containing the main functionality -- in our case, these are the components.

As mentioned in the introduction, the main idea which lead to the development of the JaBInT framework was the abstract model of the ITS architecture. From our perspective (and according to Clancey [5]), the ITS architecture usually consists of four parts. These are realized as the components: user interface component, knowledge component, (pedagogical) process steering component, and learner component (see figure 1). In JaBInT, these components are called "semantic components", as they provide the semantic frame for the system developers. Each component consists of at least one module (in most cases, there will be more). As shown in figure 1 on the right side, a module can either be a programmed functionality, a database, or for example an external simulator.

The *User Interface component* contains all modules related with the prompting of information directed to the user and receiving input from the user. The *User component* contains modules related to the user, e.g. the learner in the system. Information stored here can range from personal preferences, history and expertise to comprehensive learner models. The *Expert component* contains all technical and background knowledge, same as the pedagogical or instructional knowledge related to the content of the training system. Usually, the content of the User component will change at runtime or at least after usage of the system (e.g. what the learner has learned), whereas the content of the Expert component should not change while using the system. The *Process Steering component* is the central part of the ITS. Here, pedagogical and instructional decisions take place, and adaptation of content and navigation to the learners performance, decisions and competence is executed. The

functionality of a prototypical Process Steering component is formally described in the Tutoring Process Model [23], [24].

Modules not only implement the actual functionality of the ITS -- they are also responsible for the communication. Communication is realized via message passing between the ports. The ports can be seen in figure 1 on the right side (exemplarily skeched for the Expert compontent).

The complete ITS is described on the basis of XML files: For each module, an XML file is available, which contains information about the input and output ports and the databases. Another XML file is available for the complete ITS project. This file describes the assignment of each module to each semantic component and the interrelation of the modules (via ports). Whereas the main structure of the ITS, consisting of the four semantic components, is fixed, the modules may be exchanged or the list of modules (and thus the functionality) in a component might be extended.

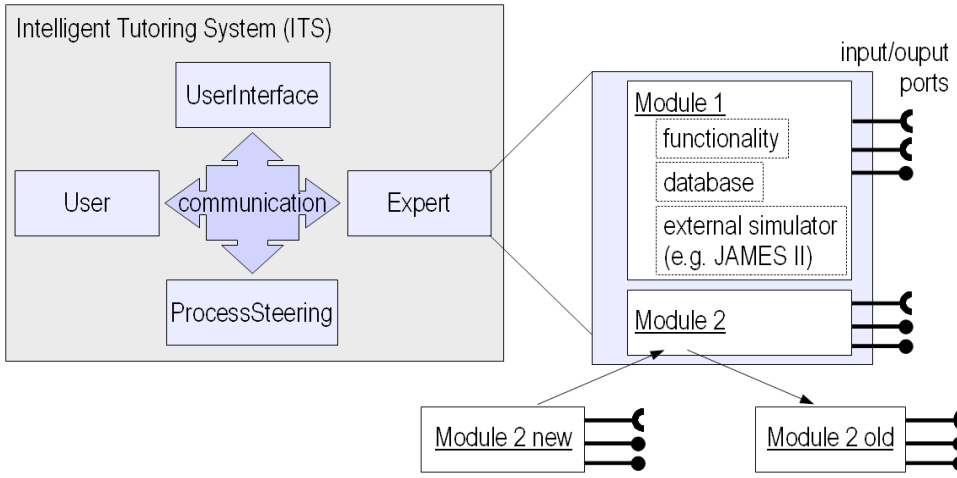

Fig.1. Architecture of the JaBInT Framework. Sketched exchange of components (left). Component with exchange of modules (right) [27].

3 Prototypical Realizations

As a first test, we implemented a small ITS, called ChemNom [16]. Herein, students of chemistry or pharmaceutics can train their knowledge about atomic structures and chemical nomenclature in the field of organic chemistry. A screenshot of the system (in German) can be seen in figure2.

The learner has a set of atoms and a set of bonds (i.e. bonds between the atoms). By drag'n'drop, the learner can take atoms and bonds and construct a chemical structure in the editor field (on the right in figure 2). The communication with the virtual tutor is shown in the lower part of the window. The tutor asks for the structure of a certain chemical element, e.g. ethene in figure 2. Based on the (chemical) expert knowledge in the Knowledge Data of the Expert semantic component (i.e. facts and rules related to

organic chemistry, like atoms, bonds related to atoms, exceptions and the like), the virtual tutor (realized as a module in the Process Steering semantic component) gives correction, feedback and help, and also tries some sort of motivation (on a textual basis). The JaBInT framework for ChemNom consists of the following components and modules (all realized in Java and XML). The *Process Steering component* contains two modules, which are the module Preparation and the module Evaluation. "Preparation" contains functionality for adaptive selection of the task to solve (adaptation to the learner's knowledge in the field and his performance) and for the presentation of the task.

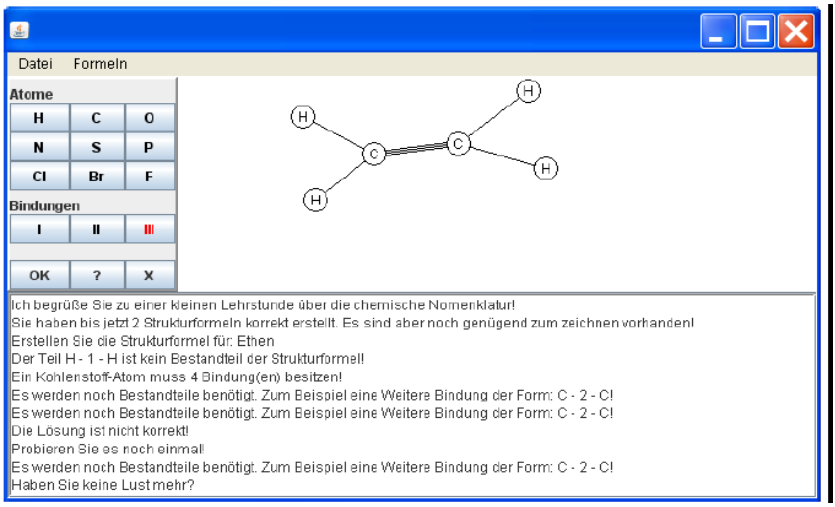

Fig. 2. ChemNom -- an ITS for teaching the chemical nomenclature (figure from [16])]

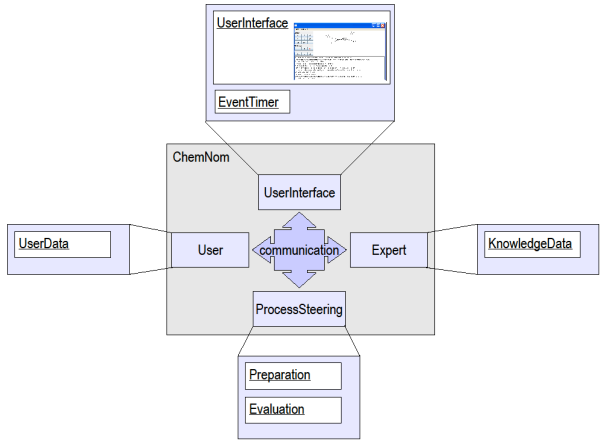

Fig. 3. Architecture of ChemNom: semantic components and modules

"Evaluation" is responsible for reacting on the learner's demands (e.g. correction or help) and for evaluating the learner's solution. The *Expert component* contains only the module Knowledge Data as an interface to the chemical facts and rules contained in the related database. Communication with the database takes place via SQL, results are displayed based on Java. The data structure is realized in a way that equivalent or similar chemical structures can be detected and correctly evaluated. The *User component* also contains only one module, i.e. the UserData module. This module contains an interface to a database -- in this case to the learner database, which contains the user's profile. The profile is a small set of information about the learner, consisting of username, number of tasks solved, number of errors in each try, number and id of already given help or hints. The *UserInterface component* contains the UserInterface module, whose output can be seen in figure 3. An additional module is the EventTime, which checks and evaluates temporal delays and inactive phases of the user.

In ChemNom, 28 unidirectional couplings between ports have been realized. For example, if the learner clicks on "help", the user interface directs this to the evaluation by using the coupling "helprequest". In [29] we describe some communication patterns (i.e. very typical couplings), which lend themselves to steer the communication between potential modules.

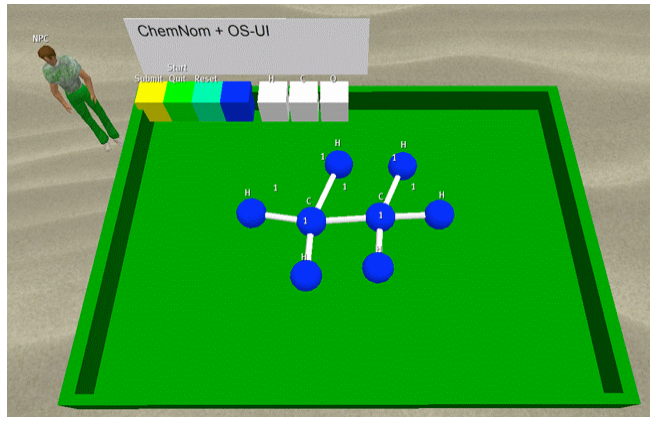

Fig. 4. Exchanging the user interface to ChemNom 3D [20]

To show that the exchange of modules is comparably easy, we started with exchanging the user interface of the classical problem-oriented ITS ChemNom. We changed the existing two dimensional (and comparably boring) user interface (shown in figure 2) to a three dimensional user interface, as can be seen in figure 4. In a first step, the changes only affected the user interface – instead of the editor-like approach, we have developed and integrated OpenSim into the framework. OpenSim is an open source version of the Second Life (Linden Labs) – and it is even more powerful with regards to educational purposes. Communication with OpenSim takes place via text messages using a HTTP server. The message exchange is embedded in JaBInT – none

of the existing modules is aware of the communication with an external source. This approach showed that exchanging of one module is possible without affecting the core structure of the ITS.

The new UserInterface module works on the same principle as the old one and sends the same data to the same other modules, even if the new module also exchanges data with an external source (the 3D world). Regarding the tutoring content, no changes have been made. But we observed one side effect: In the 3D approach, the user's avatar has to manually take up and arrange the atoms (blue balls on a table) and manually grab the bindings and to place them between the atoms. The non-player character, which is nothing more than a visual place holder of the virtual tutor in the ITS ChemNom, gives advice and help (shown on the display in the background). The side effect is that the learner's motivation has been comparably higher than in the previous 2D version and the feeling was more like "playing a game" (even if content and help have been exactly the same as in the old version). The number of users have not been high enough (neither has been the test phase long enough) to put this on a firm empirical ground, but the finding is still interesting enough to be mentioned.

After this test, another question arose: is it possible to change the core idea of the ITS without changing all modules? After our experiences with the game-like feeling in the 3D World, we changed the classical problem-oriented ITS to a pure game-based ITS. The user interface's modules are replaced by new modules. The only restriction is to somehow keep the original communication channels, i.e. the new modules must provide the same input and output ports to transmit the same data type. What is completely new, is the integration of an interactive Flash (Adobe) animation for the user interface (see figure 5). The game is based on the Tetris idea (Alexey Pajitnov, 1984). In the interface's lower part, a list of molecules can be seen. Dropping bricks contain a name of a chemical molecule, e.g. propene. The user's task is to steer the brick with the name to the correct molecule structure in the window's lower part (using the arrow keys). Color coding informs the user about his success. Bricks pile up and form stacks. Only after a number of correct answers, a row of bricks is deleted. The game is over, like in the original Tetris game, when a pile rises to its maximum height.

Fig. 5. ChemisTris -- a game based ITS for teaching the chemical nomenclature (figure adapted from [16])

The changes from the ChemNom ITS, over ChemNom3D to ChemisTris are shown in figure 6. All modules are preserved throughout the whole evolution, apart from the ones in the UserInterface semantic component.

In this version of ChemisTris, the checking of the learners "answers" is done by the Evaluation module of ProcessSteering, but the game rules are directly implemented in the UserInterface module. This was the straight forward way of implementing the very basic behavious of the game, consiting only of counting the consecutive right answers and then clear a row if completed, and increasing falling speed of the bricks with each level. However, game rules have semantically no relation to the user interface but are a set of rules determining the behaviour of the system in certain situations. Theoretically, game rules shouldn't therefore be held in the UserInterface component, but in the Expert component. The way it is currently implemented, changing game rules would mean adapting the UserInterface module, which is a much complexer and error-prone as adapting a small GameRules module from the Expert component. The correct way of implementing ChemisTris is shown in figure 7. This is a typical example of the rigorousness that the developper has to exercise while implementing a module-based system.

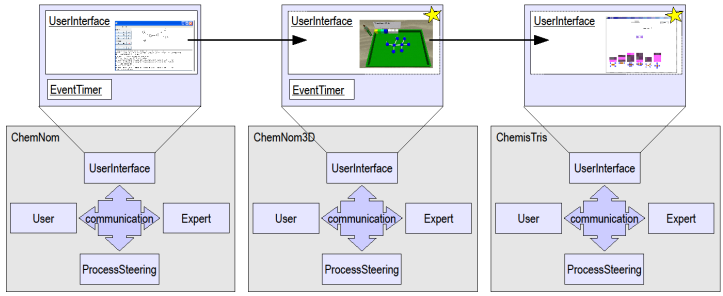

Fig. 6. Architecture of ChemNom (on the left), ChemNom3D (center) and ChemisTris (on the right). All modules apart from the ones in the UserInterface component stayed unchanged during all this transformations.

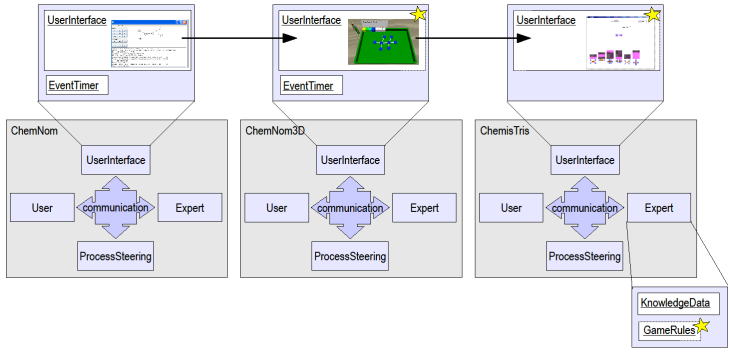

Fig. 7. Architecture of ChemNom (on the left), ChemNom3D (center) and the corrected architecture of ChemisTris (on the right). The Expert component contains in Chemistris a new module holding the game rules.

4 JaBInT-based Game Architectures

Computer and video games with educational content promise to foster intrinsic motivation in learners, enable free experimentation without having to fear bad consequences, and to facilitate transfer of knowledge to real-word problems by a highly illustrative use of virtual environments and interactive simulations. This prospect goes beyond simple quizzes or action puzzle games like ChemisTris.

For ChemisTris, it was possible to arrive at a simple drill-and-practice game by exchanging only the semantic component UserInterface (or rather, its modules). More complex cognitivist or constructivist educational scenarios demand more complex gameplay, though, and thus require conceiving a completely new piece of software architecture. We will show here – by two examples – that certain instructional approaches and technologies can be mapped onto certain game genres, which again can be mapped onto the JaBInT architecture. This opens up all the advantages of component-based software design for the field of game-based learning (GBL).

Although being largely overlooked by GBL researchers, possibly due to its complexity, the genre of computer role-playing games (RPGs) appears to be particularly suited as the basis for educational games. Genre conventions of entertainment RPGs are development of the role of a player character (PC), immersive exploration, epic story, combat, interim quests, grabbing treasure, resource management, and problem solving [7]. Today's RPGs feature many more simulated activities besides combat, e.g. casting spells, picking locks, setting traps, crafting, conversations with non-player characters (NPCs), trading, or feeding/healing.

In the GBL context, these features can support adaptivity by mapping learner characteristics to the PC role, reflection via in-game journals, combination of the traditional educational game genres simulation game and adventure game by situating adventure quests in immersive simulated environments, independence from a certain instructional approach by offering a pool of activities, and cooperative learning – in the case of massively multiplayer online role-playing games – by a variety of collaboration tools. When developing game-based ITSs, the aspect of learner adaptation is of special importance. For instance, the entertainment RPG Oblivion (Bethesda, 2006), assigns opponents based on the PC's combat skills, or hands out certain quests only if the PC is a member of a certain guild.

Developers and researchers have tried to map adaptive gameplay onto educational game architectures. However, existing approaches keep game and ITS separate, embed a separate game in an existing ITS (or vice versa), or find mappings from ITS to game only for some ITS parts. This observation led to our proposal of a fully integrated game-based ITS architecture [18] based on our Plug'n Train framework: figure 8. In this approach, ITS functionality *becomes* the game functionality.

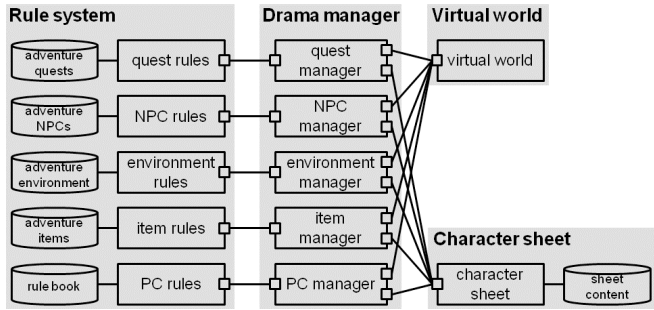

Fig. 8. Generic educational RPG architecture

The in-game Character Sheet does not only describe characteristics of the PC, but at the same time those of the learner. It is the RPG equivalent of the User semantic component. The Rule System provides access to educational quests (at the same time TEL exercises or scenarios), NPCs (also personifications of the tutor), environments (the game world), and items (in-game tools) as well as a virtual rule book specifying what the PC can be, do, and learn. The Rule System as a whole is the RPG equivalent of the Expert semantic component. The Drama Manager uses Rule System and Character Sheet data to tell interactive, educational stories adapted to the individual learner/player. This corresponds with the Process Steering component. Finally, the User Interface becomes a Virtual World, similar to the one created for ChemNom 3D (just a lot larger and with many more interaction possibilities, of course).

We are working on instantiating the generic architecture with modules for different instructional approaches and content data for different educational domains. An inquiry learning quest on population dynamics in Biology [17],[20] makes use of the crafting RPG activity, which includes environment rules (predator fish, prey fish, a simulation of predator-prey dynamics) and item rules for interactive manipulation of the simulation (fishing rods, fodder tube). An interpretative writing quest on Shakespeare's Hamlet [17] makes use of the conversations RPG activity, which includes mainly NPC rules (reactions of characters from the play when questioned about the murder of the late King).

Each RPG activity (crafting, conversations) implies a certain communication pattern between JaBInT modules, specifying the sequence of atomic actions and associated data flow. Implementing these exemplary quests and activities using JaBInT will result in the first modules of an educational RPG component library. For instance, the first implementations will have an isometric Virtual World. We can later exchange this with real-time 3D, without having to modify any of the other three semantic components (or the content data, for that matter).

In our second GBL example, the entertainment game genre construction & management simulation (CMS [28], e.g. Sim City, Maxis/Will Wright, 1989) and its specialisation tycoon game ([2], e.g. Railroad Tycoon, MicroProse/Sid Meier, 1990) are particularly close to the teaching approach of a business simulation. Typical gameplay activities in these genres are instances of acquisition (gathering and extracting resources), production (consuming resources), sale (allocating resources), management (managing profitability, directing people, handling alerts, refining old

processes), and construction (researching and implementing new processes, building and upgrading units and structures, creating paths; cf. [2]).

When reviewing literature on educational business simulations and studying a corpus of economical CMSs, we noticed a number of conventions by which these games stand out: They are turn-based, use a scoring system, run simulations, pick from the pool of the activities listed above, manage the environment (including the possibility of the player interactively constructing or changing the game world), and introduce events (e.g. catastrophes to be dealt with). This led to the definition of a second game-based ITS architecture based on JaBInT, to be realised as a growing library of modules: figure 9.

As a proof of concept, we have designed (but not yet implemented) the JaBInT modules of a bicycle factory simulation game teaching sub-domains of Economics. The major difference between the RPG and the CMS architecture is that a CMS (at least as a mapping from educational business simulations) is turn-based. Hence, it was easier to identify an overall communication pattern of the game mechanics (roughly: planning, simulation, scoring). At a lower level of abstraction, we found communication patterns for the individual activities (acquisition, production, sale, management, construction) – similar to activities in RPGs.

Having architectures for both complex educational game genres which are both based on the Plug'n Train concept and the JaBInT framework makes it possible to mix genres. For instance, Europa 1400: The Guild (4HEAD, 2002) is an entertainment game simulating trade and professions in medieval Europe by mixing CMS with RPG mechanics. Moreover, some modules of the CMS Rule System (economy rules, simulation rules) could be reused when implementing the trading activity in RPGs.

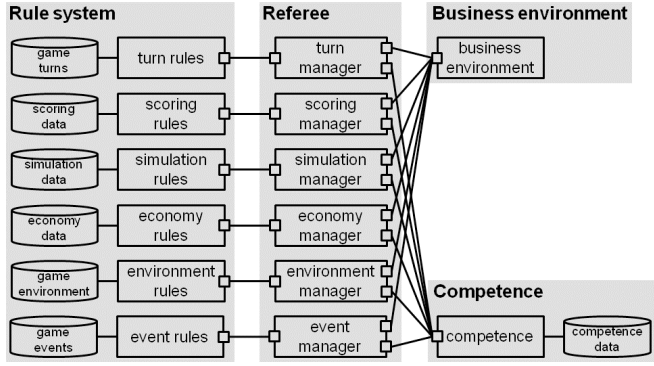

Fig. 9. Generic educational CMS architecture

In order to allow component-based development of realistic simulation-based games, we are currently connecting JaBInT to the external simulation framework JAMES II. A major challenge in this endeavour is that simulation-based games require so-called human-in-the-loop simulation. The player may change simulation parameters interactively, on the fly. The simulator must then compute results in compliance with real-time constraints. This shows that communicating modules need to fit together not only statically by sharing communication ports and being a part of communication flows and patterns, but also in their dynamic behaviour and performance during run-time.

5 JaBInT towards Ubiquitous Learning

The JaBInT framework promoting a modular architecture, it is only one step to the idea of decentralising the different components and creating a network of modules whose use is dynamically decided upon user preferences and environment parameters. Though, using JaBInT to create pervasive or ubiquitous learning environments raises the questions of compatibility between the modules and integration of the context factor.

The question of compatibility has been addressed briefly in the interchangeability of modules. In order to replace the use of one module with another one, the type of ports and of exchanged data must be maintained. As it is very difficult to keep track of all communications taking place between all the possible combinations of modules, we designed an authoring system in which communications semantically linked to the same task are regrouped. An example is shown on 0on the left, picturing the identification process. Almost every ITS needs the user to identify himself at the beginning of a session. This identification process requires the competent User Interface module to communicate the gathered data about the user, which are transmitted to the competent User Data module, returning the corresponding user identification key. The authoring system gives also interactively information about the type of data transferred over each connection. Supposing that different User Interface modules are able to connect on the system (e.g. a personal computer, a hand helded), each one is able to connect into this identification connection flow and so to induce the tasks to perform with the system. We are currently developping a library of module interfaces to implement for each specific use, as well as the corresponding communication flows.

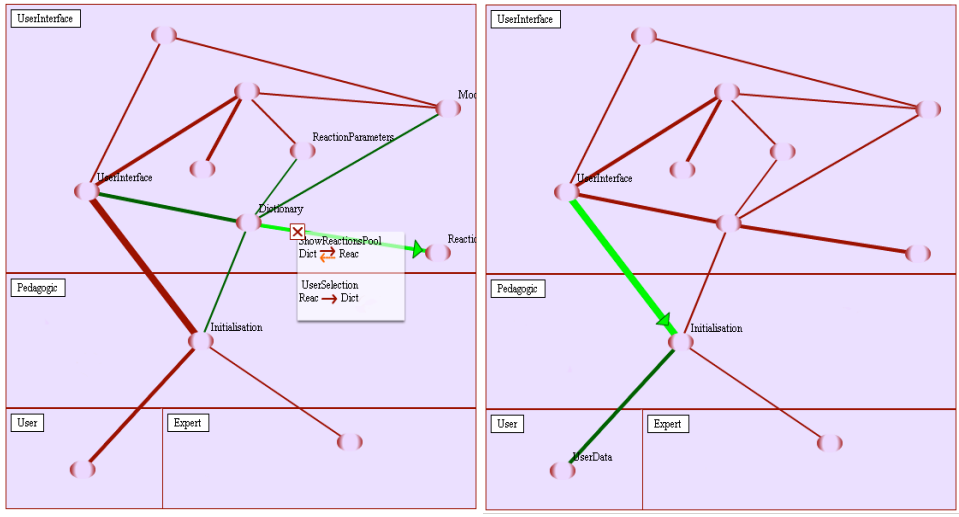

Fig. 10. Examples of paths (left: open module, right: identification)

Still unaddressed is the question of the gestion of context information. A major part of the ubiquitous computing is the adaption to the special user in his special environment. In JaBInT, the User semantic component is already designed to contain all information about the user. Few are static (e.g. name, birth date or some kind of identification key) but most are dynamic information (e.g. preferences, statistics or results). Although one could consider designing an additional semantic component to fit in the information about the user's context and all ubiquitous computing specificities, we prefere to integrate the corresponding modules into the User semantic component. It is consistent as well as convenient. Consistent because context data is directly related to the user and prone to change from one session to the other, or even within a session, like all other User content. Convenient because environment data can have direct influence e.g. on the user preferences, and beeing held in the same semantic component enables an quicker and easier adaption of an existing system to the ubiquitous aspect.

7 Outlook

In this paper, an approach to develop ITSs and "not intelligent" training systems based on a component framework is described. The implementation of the framework, called JaBInT (Java Based Intelligent Tutoring) has had its roots in research related to ITS patterns and to formal models for ITSs. However, the world has changed in the last ten to five years -- topics of ubiquitous and pervasive learning are state of the art in teaching and training system research, "adaptation" is used more often than "intelligence". Our main idea about future development in game-based learning systems and in other eLearning systems (e.g. Intelligent Tutoring Systems) is that the careful design and engineering of these systems has to continue -- also or moreover if liquid learning places are addressed. Developing teaching and training systems is a very demanding task – as the developer has to be aware of human factors, of learning preferences, of technical devices, of different instructional approaches and of different content-related requirements. On one hand, we can look back on a comparable long history of system development. Thus, we should have a lot of experiences. When looking at the developments in the last years, a lot of approaches which re-invent the wheel can be found. Thus, on the other hand: how have we learned from this long tradition? How can well known insights be transferred to the new "mobile" world of computer based teaching and training?

Our approach takes up a claim, which is not new: softare systems have to be carefully engineered rather than invented ad hoc. A component based design, like the one suggested in our paper, can help in the development of flexible, visionary systems, which might be the basis of liquid learning places. Based on a clear software design, is might be easier to define aspects of liquid learning, for example: where is the learner, who is the learner, what are his interests at the moment, and which mobile device is he using in which place. A component based system might provide enough flexiblity to allow for adapting to the learner, the learner's location and the currently used device. However, this is still rather at the level of "fiction" than at the level of science.

More important for development of teaching and training systems in future are two other aspects: a component based design can help to compare systems from the

software engineering point of view, and, last but not least, it helps to develop comparable systems for scientific empirical evaluation.

Acknowledgements. This article describes the combined efforts reached by two projects and the work of several students. Special thanks in this context got to Alexander Neef, who finished his master thesis in the context of JaBInT. The authors would like to thank the DFG (German Research Foundation) and the EU South Baltic Programme for financial support of parts of this project.

References

1. Anderson, J.R., Boyle, C.F., Corbett, A.T. and Lewis, M.W.: Cognitive Modeling and Intelligent Tutoring. Artificial Intelligence, 42, pp. 17--49. (1990)
2. Aldrich, C:. The Complete Guide to Simulations and Serious Games: How the Most Valuable Content will be Created in the Age beyond Gutenberg to Google. San Francisco, CA: Pfeiffer, (2009)
3. Bergin, R. A., Fors, U.G.H.: Interactive Simulated Patient - An Advanced Tool for Student-activated Learning in Medicine and Healthcare. Computers and Education, 40, 4, pp. 361--376. (2003)
4. Corbett, A.T., Koedinger, K.R., Anderson, J.R.: Intelligent Tutoring Systems. In: Helander, M., Landauer, T.K., Prabuh, P. (Eds.): Handbook of Human-Computer Interaction, pp. 849--874. Elsevier Science B.V., 2nd completely revised edition (1997)
5. Clancey, W. J.: Methodology for Building an Intelligent Tutoring System. In: Kintsch, W., Miller, J.R., and Polson, P.G. (eds.): Methods and Tactics in Cognitive Sciences, pp. 51--84. Lawrence Erlbaum Associates, Hillsdale, New Jersey, London, (1984)
6. Clancey, W. J.: Knowledge-Based Tutoring - The GUIDON Program. The MIT Press (1987)
7. Hallford, N., Hallford, J.: Swords & Circuitry: A Designer's Guide to Computer Role-Playing Games. Prima Tech, Roseville, CA (2001)
8. Harrer, A., Martens, A.: Towards a Pattern Language for Intelligent Teaching and Training Systems. In: Proc. of the 8th International Conference on Intelligent Tutoring Systems. LNCS vol. 4053, pp. 298--307. Springer. (2006)
9. Harrer, A.: Software Engineering Methods for re--use of Components and Design in Educational Systems. International Journal of Computers & Applications. Special Issue on Intelligence and Technology in Educational Applications, 25, 1. Anaheim, CA (2003)
10. Himmelspach, J., Uhrmacher, A.M.: Plug'n simulate. in: Proceedings of the 40th Annual Simulation Symposium, pp. 137--143. IEEE Computer Society (2007)
11. Illmann, T., Weber, M., Martens, A., Seitz, A.: A Pattern-Oriented Design of a Web-Based and Case-Oriented Multimedia Training System in Medicine. In Proc. Of the 4th World Conference on Integrated Design and Process Technology, Dallas, US (2000)
12. de Jong, T., van Joolingen, W.: Discovery Learning with Computer Simulations of Conceptual Domains. Review of Educational Research, 68, pp. 179--201. (1998)
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA (1995)
14. Kuenzel, J. and Hämmer, V.: Simulation in University Education: The Artificial Agent PSI as a Teaching Tool. Simulation -- Special Issue on Modeling and Simulation in Teaching and Training, 82, 11, pp. 761 -- 767. SCS (2003)
15. Lelouche, R.: Intelligent Tutoring Systems from Birth to Now. In: KI-Kuenstliche Intelligenz, 4, pp. 5--11. AKA, IOS Press (1999)

16. Maciuszek, D., Ruddeck, G., Weicht, M., Martens, A.: Plug 'n Train fuer Game-based Learning. In: Proc. of the Workshop Game-based Learning (at the DELFI 2009), Logos Verlag, Berlin (2009)
17. Maciuszek, D., Martens, A.: Computer-Rollenspiele als didaktikübergreifendes Lernspiel-Genre. In: Prod. of the 4th Workshop Game-based Learning, pp. 43-57, Stuttgart, Fraunhofer-Verlag (2010)
18. Maciuszek, D., Martens, A.: A Reference Architecture for Game-based Intelligent Tutoring. In: Felicia, P. (ed.) Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches, IGI Global, Hershey, PA (2011 (to appear))
19. Maciuszek, D., Martens, A.: Patterns for the Design of Educational Games. In: Edvarsen, F., Kulle, H. (eds.)Educational Games: Design, Learning and Applications, chap. 8. Nova Science Publishing, Hauppauge, NY, USA (2010)
20. Maciuszek, D., Ruddeck, G., Martens, A.: Component-based Development of Educational Games: The Case of the User Interface. In: Proc. of the European Conference Game-based Learning (2010)
21. Martens, A., Bernauer, J., Illmann, T., Seitz, A.: Docs 'n Drugs - The Virtual Polyclinic. In: Proc. of the American Medical Informatics Association Conference 2001, pp. 433--437 (2001)
22. Martens, A, Cap, C.: Patterns for eLearning Systems. In: Proc. of the eLBa 2009 e-Learning Baltics International Scientific Conference, pp. 115--124. Fraunhofer IRB Verlag (2009)
23. Martens, A., Uhrmacher, A. M.: A Formal Tutoring Process Model for Intelligent Tutoring Systems. In: Mantaras, R. L. & Saitta, L. (ed.) In Proc. of the 16th European Artificial Intelligence Conference ECAI, pp. 124--128. (2004)
24. Martens, A.: Time in the Adaptive Tutoring Process Model. In: Mitsuru Ikeda, K. D. A. T. C. (ed.) Proc. of the 8th International Conference on Intelligent Tutoring Systems 2006, pp. 134--143. Springer (2006)
25. Mayo, M., Mitrovic, A., McKenzie, J.: CAPIT: An Intelligent Tutoring System for Capitalization and Punctuation. In: Proc. of the Int. Workshop for Advanced Learning Technologies pp. 151--157 (2000)
26. Michaud, L., McCoy, K., Pennington, C.: An Intelligent Tutoring System for Deaf Learners of Written English. In: Proc. of the 4th International ACM Conference on Assistive technologies. ACM Press, New York (2000)
27. Oertel, M., Martens, A., Himmelspach, J.: Teaching and Training System Plus Modeling and Simulation - A Component Based Approach. In: Proc. of the EUROSIM/UKSim, pp. 475--480. IEEE Computer Society Conference Publications, CPS. (2008)
28. Rollings, A., Adams, E.: Andrew Rollings and Ernest Adams on game design. Indianapolis, In: New Riders (2003)
29. Ruddeck, G., Martens, A.: Communication Patterns in Component-Based Intelligent Tutoring Systems. In: Proc. of the IEEE International Conference on Advanced Learning Technology ICALT. IEEE Press (2010)
30. Schuemmer, T., Lukosch, S.: Patterns for Computer-Mediated Interaction. John Wiley & Sons (2007).
31. Szyperski, C.: Component Software. Addison Wesley, Component Software Series, ACM Press, New York (2002)