Journal of Universal Computer Science, vol. 13, no. 4 (2007), 504-530
submitted: 25/10/06, accepted: 23/4/07, appeared: 28/4/07 © J.UCS

Formal Representations of Learning Scenarios: A Methodology to Configure E-Learning Systems

Denis Helic
(Institute for Information Systems and Computer Media
Graz University of Technology, Austria
dhelic@iicm.edu)

Abstract: Nowadays, advanced E-Learning systems are generally pedagogy-aware. Commonly, these systems include facilities for defining so-called learning scenarios that reflect sophisticated pedagogical approaches such as collaborative writing or project-oriented learning. To support different learning activities from such scenarios the technological infrastructure of these systems must be appropriately adjusted and configured. Usually, this configuration process is laced with a number of difficulties. Most of these difficulties are caused by the fact that scenario capturing is achieved through informal user-developer dialogues. Typically, the result of such informal dialogues contains inconsistent and incomplete information because of misunderstandings and the complexity of the interactions within a scenario. Consequently, the configuration of the system is suboptimal and a number of iterations are required in order to achieve better results. In this paper an approach to improve this situation is presented. This approach is based on a general formal representation model for describing learning scenarios. A particular formal description of a concrete learning scenario is obtained through a user dialogue with a wizard tool. At the next step, this formal description might be automatically processed to facilitate configuration process. The paper is concluded with some experiences gained by applying this approach in two E-Learning projects.

Keywords: E-Learning, E-Learning system, learning scenario, formal representation, configuration
Categories: H.1.0, H.4.0

1 Learning Scenarios in E-Learning

It is our experience that successful E-Learning projects always concentrate on pedagogical aspects to enable efficient technology-enhanced knowledge transfer. As such, these projects are far less focused on the technology itself [Hirumi 2002; Hobbs, 2002; King and Puntambekar, 2003; Leasure et al., 2000; Mioduser et al., 2000; Oliver et al., 2002]. For example, one of such projects was the project called CORONET (Corporate Software Engineering Knowledge Networks for Improved Training of the Work Force) that was funded by the European Commission (IST-1999-11634). The main purpose of the project was to analyze, implement and evaluate a number of tools for support of collaborative knowledge transfer processes. Each of such tools utilized the current and advanced Web technology to facilitate and spread the flow of knowledge from people possessing that knowledge to people who needed to acquire it by following a particular collaborative pedagogical approach. Thus, processes such as Web-based tutoring, Web-based knowledge mining, Web-based collaborative writing, and collaborative project-oriented learning have been

supported. The evaluation of the project results with respect to the increase of learning effectiveness by knowledge sharing and collaborative learning generally indicated improved learning effectiveness [Pfahl et al., 2004; Helic et al., 2004; Helic et al., 2005].

One simple strategy to utilize pedagogical aspects in E-Learning is based on managing so-called learning scenarios. This strategy was successfully applied in a number of E-Learning projects at the University of Technology Graz [Helic, 2006; Helic and Durco, 2005; Ebner et al., 2005, Dreher et al., 2004]. In those projects a learning scenario was defined as a combination of the following components:

- A particular way (i.e. a story) of working with the system to achieve a particular learning goal. Typically, the story was represented as a collection of learning activities that need to be carried out to accomplish the goal.
- The user roles that are involved in the story, e.g. teachers, tutors, students, or learners.
- The system tools, features and services that are needed to support the activities.
- The educational content relevant to the learning goal.

Using this scenario structure, scenarios such as thematic uploading, thematic discussion, goal-oriented, or reflective learning were developed in these projects. Further, one of the important requirements in these projects was a possibility to introduce new learning scenarios and customize already existing ones. The systems and technological infrastructure fulfilled that requirement by providing a generic learning scenario framework. The framework utilized extensibility and customizability by providing flexible system configuration possibilities. Thus, supporting a new learning scenario or customizing an existing scenario was equivalent to providing a new system configuration or adjusting an existing configuration, respectively.

However, such a general and flexible pedagogy-aware framework is inevitably technically complex. Therefore, the process of configuring the framework to reflect the scenarios is related with a number of problems. Typically, these problems are visible at different stages of such a scenario management process.

Capturing of scenarios is tedious and error-prone. It is carried out in informal settings through user-developer dialogues. The results of such dialogues are scenario requirements documents in a narrative form. Typically, this final result contains inconsistent and incomplete information because of misunderstandings or the complexity of the interactions within the scenario. Basically, this problem is equivalent to a general user requirements analysis problem in any software and system development process [Meyer, 1985; Jarke et al., 1998].

Technical realization, i.e. configuration, and customization are extremely difficult because of the rich and complex functionality that is needed and because of the heterogeneity of the technological environment in question. Firstly, in order to support sophisticated learning activities and scenarios the system must provide sophisticated functionality. That functionality is typically organized in highly complex structures and processes. For example, collaborative writing scenario includes activities such as writing, reading, reflecting, commenting, and discussion. These activities can be mapped onto the system functionality in the following way.

The discussion activity is mapped onto a discussion forum. The writing activity is mapped onto an editor tool with upload functionality. Additionally, the editor is associated with a version control system to control different versions of the documents. The reflecting and commenting activities might be mapped onto an annotation tool, and all of the activities are connected to the user authentication and user rights module that decide who can do what. Thus, even this simple scenario example can lead to a complex development and configuration process. Moreover, E-Learning systems are typically developed in a Web-based technological environment, which is a very fast evolving and ever-changing environment. Obviously, such a dynamic environment increases the complexity of the development and configuration process of E-Learning systems.

We believe that one possibility to remedy these problems is by developing formal specifications of learning scenarios. Such formal specifications would meet two goals; firstly, inconsistency and incompleteness of a captured scenario description can be partially or completely avoided; secondly, strategies and automatic or semi-automatic procedures for supporting system configuration and customization can be developed on the top of that formalism. As a consequence of these two goals formal specifications must be:

- Easy to understand and use for all, i.e., the users of the system (e.g., teachers, tutors, students, etc.) as well as the developers of the system.
- Sufficiently expressive, yet still simple enough as to allow semi-automatic or automatic processing and analysis in order to obtain at least a first prototype of the system at different levels of abstraction and granularity, e.g., architectural model, component-model, or even automatically generated configuration scripts or implementation and integration code if needed.

The idea of formal specification of the user requirements (user scenarios, use cases) is by no means a new idea in software engineering. For many years now developers and researchers alike have been trying to apply formal methods in software development in general and user requirements engineering in particular [Meyer, 1985; Clarke and Wing, 1996, Hong and Lingzi, 2000]. However, there exist a number of difficulties related to formal methods in developing information systems caused by fast changing requirements, user bases, usability and design issues, or rapidly changing technologies [Land and Hirschheim, 1983; Avgerou, 1987; Overmyer, 2000; Bolchini and Paolini 2004]. Moreover, such formal methods are typically applied only for validation and checking purposes rather than as a system configuration support [German, 2000].

One of the possibilities to reduce the complexity of these problems is to use domain-specific knowledge in developing a formal representation model. The domain knowledge can be used not only to support capturing of the user and scenario requirements but also for automatic or semi-automatic processing and analysis of the collected scenarios by utilizing the domain-specific semantics. This approach is very similar to a modern software engineering approach called Model Driven Architecture (MDA) first proposed by the Object Management Group (OMG). According to this design approach a system specification is developed in the form of a formal domain model that can be processed to automatically generate parts of implementation, integration, or system configuration code [OMG, 2003].

2 Management of Learning Scenarios

One of the basic problems in capturing scenarios is a mutual lack of knowledge about the technical and pedagogical aspects on the side of the users (teachers) and system developers, respectively. On the one hand, the users of the systems lack the system expertise and experience, i.e. they have a particular pedagogical approach in mind, but they do not know if and how that pedagogical approach or at least part of it is technically feasible. On the other hand, the system developers have limited understanding of the domain, the subject matter, and the pedagogy involved in a learning scenario. Thus, there exists a so-called "impedance mismatch" between the pedagogical aspects of a learning scenario and its technical realization. In other words, both the system developers and users are confronted with input from the other side that is typically inadequate to them. Let us illustrate this problem with two real-life examples from some of our recent E-Learning projects.

2.1 Problems with Scenario Management: Case Study 1

The first project in question is Ephras [Helic, 2006; Helic and Durco, 2005], a project that has been funded by the European Commission under the Socrates/Lingua2 programme (117024-CP-1-2004-1-SI-LINGUA-L2). The goal of the project was to develop a computer supported phraseology learning material for students of foreign languages for four European languages - German, Slovak, Slovenian and Hungarian language.

One of the E-Learning components developed within the project was a module with 150 interactive tests for selected phrases in those four languages. Those tests served to check understanding and knowledge of the phrases in question, as well as for improvement of skills for producing these phrases in a written and spoken foreign language. As such they followed a sound pedagocial approach that has been developed by the pedagogical partners (foreign language teachers) in the form of an exercise typology. This exercise typology was a result of expertise and experience of the teachers collected in a number of years in teaching phraseology in traditional classroom settings. Additionally, the questions and tests have been aligned according to language, topic, knowledge, and skill level of the students.

On the technical side, IMS Question & Test Interoperability (QTI) [IMS QTI, 2005] specification has been applied to develop the exercise module. The QTI is a de facto standard for cross-platform representation of questions and tests in Learning Management Systems (LMS) and there are freely available tools for authoring, running, or result processing for QTI tests. While developing the exercise module we have experienced the following problems caused by the above mentioned "impedance mismatch" between the QTI standard and its concrete application in a specific subject area such as foreign language teaching. The mismatch is visible at two levels.

First, the QTI standard is a technical specification which supports development of questions and tests for various subject domains where the questions and tests are interoperable at the level of authoring tools, question/test databases or LMS. As such the QTI standard defines a number of general question types that might be applied in a number of areas and does not take into account specific question and test types of a particular domain [Milligan, 2003]. However, there exist a number of specific

question types which are commonly used in foreign language teaching but are not reflected in the QTI standard, with crossword puzzles being only one typical example.

Second, the QTI standard is not concerned with pedagogical issues and, as a matter of fact, it tries to be as pedagogy neutral as possible [Smythe and Roberts, 2000]. Yet the basis for development of the interactive tests in the Ephras project is a sound and successful pedagogical approach for teaching phrases in foreign languages. This approach includes learning activities such as phrase recognition, phrase meaning or phrase pragmatics identification, phrase form and grammar understanding, consolidation and reflection. Each of these activities is typically represented by a number of exercise types which are used in accordance with the current context, as well as the student's knowledge and skill level. For example, identifying phrase meaning can be realized with a multiple choice question where the student needs to select the correct meaning of a phrase from a number of possible answers, with a short text essay where the student needs to write down the meaning of the phrase, with a combination of these two question types, or with a simple drag-and-drop exercise where the student needs to correlate the phrase meaning with a particular graphical representation of that phrase (see Figure 1).

Figure 1: Drag-and-drop exercise to support phrase meaning identification

Basically, development of scenarios was an iterative process with a number of repeating steps. The whole process required collaborative efforts on the side of the system developers and the teachers. The iteration steps of this development process included:

- Initial discussion between the system developers and teachers. The main goal of this session was to achieve a better understanding of both the pedagogical requirements for questions and tests and the technical possibilities offered by the QTI standard. Thus, the teachers explained the types of questions and tests that are needed, whereas the system developers explained the features of the QTI standard.
- Preparation of several QTI samples to illustrate the possibilities of the QTI standard. The prepared samples already dealt with the questions and tests from the subject matter.

- Agreement on a text-based informal format for defining the questions and tests by the teachers. Such an informal format was needed because the teachers were not familiar with creation of formal specifications, i.e. they had a non-technical background.
- Realization of scenarios by means of QTI standard.
- Test and improvement phase.

Note that some of the steps have been repeated a couple of times to obtain optimal results. Thus, the sheer amount of the development steps, as well as the amount of the work needed for communication, testing or improvement resulted in tedious work. Moreover, because of misunderstandings, communication problems, or implementation difficulties the whole development process was error-prone.

The difficulties of the scenario capturing process and their technical realization can be summarized as follows:

- Learning curve for the QTI standard for the teachers was very steep. There are no sufficient manuals, help files, tutorials or tools that would decrease the time needed to learn the standard.
- Informal nature of format for defining the needed scenarios leads to misunderstanding problems between the teachers themselves. The consequences of such problems become even more serious with an increasing number of teachers, because format disparity between different teachers also increases.
- Realization of scenarios using such an informal format is also very difficult because of misunderstanding problems between the authors and the systems developers. Again, the disadvantages become more visible as the number of teachers increases.

2.2 Problems with Scenario Management: Case Study 2

The second project in question is iVISiCE (interactive Visualizations in Civil Engineering), an E-Learning project to support the study of Civil Engineering at Graz University of Technology [Ebner and Holzinger, 2002]. Originally, the aim of the project was to investigate the possibilities of applying Web technology in education of structural engineering. Due to the fact that civil engineering students need to obtain intuitive understanding of structural behaviour the pedagogical approach is strongly based on visualizations. In addition, communication and interaction among all participants complete the learning scenario. Consequently, a great number of Web based animations, visualizations and interactive learning objects have been developed to visualize and to simulate highly complex processes [Ebner and Holzinger, 2003]. This content was implemented in a course management system, which has been made accessible by the computing department of the University of Technology Graz. The combination seemed to be quite successful for the first time. Subsequently, the gap between the sophisticated and up-to-date content [Holzinger and Ebner, 2003] and an obsolete, rigid content delivering platform became bigger and led to a dissatisfaction of end users, teachers and students alike. Finally, the course support team decided to follow a new paradigm where the content delivering system can be adapted and upgraded in the same way as content complexity and user engagement are growing

up. For that purpose a novel E-Learning platform called WBT-Master, developed at the University of Technology Graz, has been chosen [Ebner et al, 2005].

To fulfill the requirements of the iVISiCE project a WBT-Master component for management of so-called training objects has been applied (http://coronet.iicm.edu) [Helic et al., 2004]. A typical example of such training objects is a simple learning unit (i.e. a number of documents combined into a reusable, navigable collection), but this is only a single simple type of available training objects. Discussion forums, chats, quizzes, virtual laboratories, or project management rooms are also training objects which can be managed in WBT-Master. Each training object implements a particular training paradigm. For example, students are supposed to answer questions if they use an "examination room"; upload and discuss reports, if they use a "thematic discussion"; follow online/offline step-by-step explanations of most difficult topics in a “tutoring session” or in a “mentoring session”; or develop a project in collaboration with colleagues, if they use a "project management room". Additionally, training objects can be combined into a new single entity called "training course". A training course is just a combination of training objects selected for a particular study that provides also some additional communication, collaboration, and user administration tools such as evaluation, or grading tools [Helic et al., 2005a].

If we compare WBT-Master with existing E-Learning solutions one essential advantage can be recognized and that is the system’s flexibility and customizability: these features are just as desirable for any other E-Learning solution but they are an inherent part of the WBT-Master training paradigm. Any teacher selects just a few components and combines them into a new training course to provide a required training curriculum and system functionality [Ebner et al., 2005].

The new pedagogical paradigm in the iVISiCE project supported by means of WBT-Master was based on the assumption that although learning is an active cognitive process on the part of the learner, it is also a social process and develops through conversation [Motschnig-Pitrik and Holzinger, 2002]. Therefore, interaction, participation and communication have been recognized as crucial elements for iVISiCE, in which the learning environment has to enforce the possibility of community building. It should be pointed out, that the aim of this new paradigm was to compensate for the lack of communication features. More precisely, the problem was the missing communication tools developed for a specific user group – students of Civil Engineering. Thus, a new training course has been developed. Besides content management features and user management facilities this training course combined a number of communication tools such as [Ebner et al., 2005]:

- A discussion forum for writing contributions in a structured manner.
- A Web-based whiteboard for synchronous textual and vector graphic based communication.
- A thematic upload tool to help students upload, manage and discuss their practical examples for passing the examination. Furthermore, the tool assists the teachers regarding the correct time management, i.e. upload was not possible after a predefined time limit (see Figure 2).
- FAQs for exporting frequent questions from the discussion forum and arranging them into an easily navigable FAQ taxonomy.

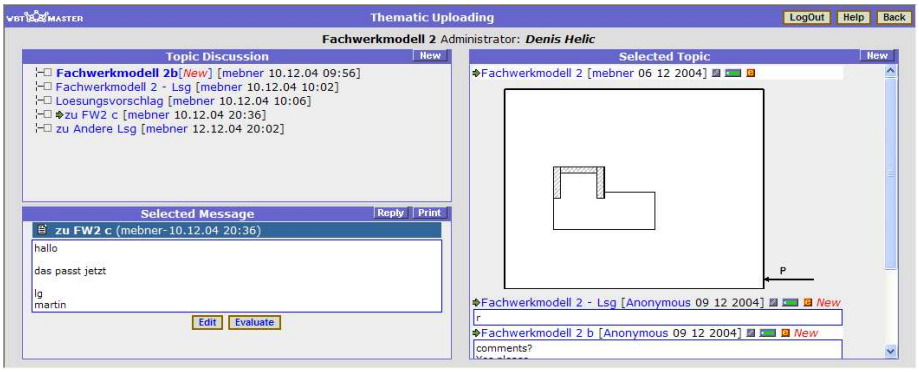

Figure 2: Thematic upload tool for management of student examples

Again, the development process was iterative and span over a number of terms and student groups with gradual improvements in the pedagogical approach, but also in the technical infrastructure needed to support that approach. The typical steps of the development process included:

- Initial discussion between the system developers and teachers, again to achieve a better understanding of both the pedagogical requirements, as well as technical possibilities offered by WBT-Master.
- Preparation of a first prototype in the form of a training course to illustrate the possibilities of WBT-Master.
- Test and improvement phase.
- Deployment and production phase, where the prototype was constantly improved by adding new training objects, customizing features, or adding new functionality.

Basically, all of these steps have been iterated at all times, with the current result achieved after three school years, two different courses, and a production phase with more than 200 Civil Engineering students.

In this project, the difficulties of the scenario capturing and realization process are similar to those experienced in the Ephras project: a steep learning curve for the system in question; the informal nature of scenario capturing; and complexity of the pedagogical approach - all lead to misunderstanding problems, incomplete, and inconsistent information.

There is also another difficulty characteristic for the iVISiCE project. Since the pedagogical approach used in the project was based on interaction, communication, collaboration, community building, and other social aspects of learning, the learning environment in question exhibited a highly dynamic nature. In such an environment the requirements are never stable and change on a regular basis. As a consequence, the technical infrastructure is in a constant “beta” state, similar to social applications typical for Web 2.0 platforms, and needs to be improved, adjusted, and customized very frequently. This deepens the problem not only for the system developers who need to react to ever-changing user requirements, but also for the teachers who manage the student community, and need to provide almost immediate feedback to

students' request. In many cases, especially when the teachers are already experienced users of the system and understand the possibilities offered by the system such problems are easily handled by the teachers themselves. For example, the teachers might demonstrate to the students how to use a particular feature in a proper way, point to a system component which might be used in a particular learning situation, or ask the system developers to extend the system functionality if there is a need for this. However, in the situation where the teachers do not posses sufficient system expertise (i.e. in the initial learning sessions) this can lead to serious problems. On the one hand, the teachers might reject an excellent student idea for improvement of the learning environment because they do not believe that the idea is technically feasible. On the other hand, they might accept an improvement suggestion for which the technical realization might be very difficult or even not possible at all.

2.3 Problems with Scenario Management: Conclusions

In both of the case studies scenario management process was related with a number of difficulties. Besides the common issues related to the user requirements engineering such as misunderstandings, incomplete and inconsistent information, or complexity of the domain (e.g. sophisticated pedagogical approaches), the dynamics of a socially-aware collaborative learning environment in the second case study introduced another level of complexity in the process. In both cases, such issues could be only resolved through an iterative development and management process over a longer period of time. Thereby, at each new iteration step a new prototype has been introduced that represented a slightly closer approximation of what the users really wanted.

These two case studies demonstrate very clearly that scenario management in E-Learning is *inherently complex*, i.e. the difficulties exist at *all levels of abstraction or granularity* of a learning scenario. For example, from the pedagogy point of view in the iVISiCE project we have been working on a higher abstraction level, i.e. the teachers defined a number of learning activities for their students and the system developers integrated a number of tools into a training course to support those activities. The activities included structured discussion, chat, drawing, or preparing and uploading materials. The teachers did not go into details of these activities because in this case the details did not have pedagogical relevance (except for preparing and uploading of materials where details such as deadlines, format, quantity and quality of material have been defined). On the other hand, in the Ephras project the teachers worked on a lower pedagogical abstraction level, i.e. they defined a single learning activity (exercises) but with a finer granularity. However, in both cases the difficulties have been present in capturing and technically realizing learning scenarios. Also, complexity of the management process was inescapable in both cases.

Consequently, if we want to deal with more complex learning scenarios that work at different pedagogical abstraction levels with a fine granularity at each of these levels the overall complexity of the learning scenario management process grows. For example, suppose that in the iVISiCE project the teachers wanted to conduct an online examination using a question-test module from the system. For instance, the examinations could follow a common examination strategy from traditional settings in Civil Engineering education and include multiple choice questions, fill-in-blank, free essays, calculus problems, or drawing and designing engineering solutions. Each

of these question types might require additional properties to be defined, e.g. for a fill-in-blank question the teachers might need to define how many blanks exist or how many positions each blank has. Obviously, capturing and realizing these scenario requirements is very similar to the Ephras project, i.e. we need to deal once more with the same difficulties at this lower level of abstraction.

Such difficulties are by no means new in the software engineering field and have been recognized previously. For example, Brooks states that "the complexity of software is an essential property, not an accidental one" [Brooks 1987]. As Booch points out the main reason for that complexity is the complexity of the problem domain [Booch 1993]. Obviously, such a complexity is inherent in pedagogy-aware E-Learning systems. A common strategy for tackling such difficulties in complex pedagogical scenarios, as well as in software development is to hide irrelevant details by working at an appropriate abstraction level and only switch to the next abstraction level or to a finer level of granularity when there is a need for it. Therefore, each technical solution for the scenario capturing and realization problem needs to follow a similar strategy, i.e. it needs to provide a mechanism for hiding irrelevant details as well as a mechanism for switching between different abstraction and granularity levels.

3 Representation of Learning Scenarios

As a solution to the above mentioned difficulties a formal representation model has been developed. The representation model comprises three components: a meta-model that defines the modeling elements and their semantics; domain models that represent particular domains and that are developed using the elements of the meta-model; a formal binding that defines an XML binding to export domain models. On top of this representation two tools have been developed: a GUI tool that supports capturing of a particular learning scenario by displaying the elements of a domain model in a user-friendly manner and an automatic processing component that processes captured learning scenarios to infer a corresponding system configuration. Both of these components work with XML-based representations (that are valid XML documents in respect to the defined XML binding) of domain models and learning scenarios.

3.1 Meta-model

The above discussion shows that the scenario capturing and realization difficulties exist at different levels of abstraction and granularity. Thus, the first requirement for the meta-model is a possibility to represent these different abstraction and granularity levels. Furthermore, to reduce the complexity of a domain model it should be possible to hide irrelevant details of that model as the need arises.

Object-oriented representation seems to be a perfect fit for such a meta-model because it is a general-purpose domain representation mechanism. The basic elements of object-orientation include objects, attributes, classes, instantiation, encapsulation and details hiding, structural relations (i.e. specialization, generalization and composition), inheritance, and associations. Let us look now at these basic

modeling elements in detail to investigate how they fit into the above defined requirements.

Objects are used to represent concepts or real-life entities. Objects typically have one or more *attributes* that represent the current state of an object. Similar objects belong to a *class* of objects that defines all attributes that these similar objects might have. Usually, we speak of objects as *instances* of a particular class that *encapsulate and hide* their current state in the form of attributes. The instantiation property is typically represented by means of an "is-a" relation. The encapsulation and hiding mechanism is a very important concept that allows us to hide irrelevant information from the rest of a domain model and only dive into the details if there is a need for it. In addition, objects and classes are typically related with other objects and classes.

With *structural relations* we can express domain knowledge with differing levels of abstraction and details. There are three typical structural relations in object-oriented modeling: specialization, generalization (which can be comprehended as an inverse relation of specialization), and composition.

Specialization reflects a so-called "is-a-kind-of" relation between two domain classes and states that a particular class is a special case of another class. For example, the QTI standard defines questions of type multiple-choice - it is a question type where students can select one or more answers from a number of answers to a particular question. Further, the standard defines a number of special cases of multiple-choice questions regarding the format and type of the answers – there can be only textual answers (text multiple-choice), image answers (image multiple-choice), or hotspot multiple-choice where students can select a particular area of an image as an answer.

Composition reflects a so-called "is-a-part-of" ("part-whole") relation between two objects. For instance, the QTI standard defines a notion of a test or a quiz, which is a collection of questions of different types, i.e. the questions are related with an "is-a-part-of" relation with the quiz.

Through the mechanism of *inheritance* we can further utilize specialization relation. Inheritance means that a special case of a class inherits all the properties (attributes and relations) of the general class. Further, the special case can refine and redefine these properties if needed. For example, typical attributes of a multiple-choice question include the question, the number of possible answers, the number of correct answers, the number of answers that students might select, and the answers themselves. One special case of a multiple-choice question is a standard true/false question, where all attributes of a general multiple-choice are inherited but the special case sets the values for those attributes by a definition. Thus, the number of possible answers is two, the number of correct answers is one, the number of answers that students might select is also one, and the answers are "true" and "false". The structural relations, i.e. specialization and composition and the inheritance mechanism allow us to create domain models at different abstraction and granularity levels. We can utilize these different levels by means of a clever user-interface that will support users in working on a single level of abstraction or granularity, and only going into details (a lower level of abstraction or a finer granularity level) when there is a need to describe these details as well. Such a user-interface would reduce the complexity of scenario capturing process to a great extent.

By means of *associations* we might express relations between objects and classes across the whole model, i.e. across the different levels of abstraction, across different levels of granularity, as well as across compositions. Let us here investigate shortly an example from WBT-Master. Suppose that a special training course should be created supporting a collaborative writing pedagogical approach. In a typical collaborative writing scenario students write and comment on a series of documents. The teacher might provide feedback for the students. In the next step, the learners work on improving the documents. At the highest abstraction level (at the pedagogical level) a model of the scenario might include the following student activities: writing, reading, commenting, reading feedback, improving results. However, we still need a technical infrastructure that will support these activities in a learning environment. Thus, that technical infrastructure needs to be reflected in the model and related to the pedagogical abstraction level. Obviously, the model of the technical infrastructure is on another abstraction level and we need to relate objects and classes between these two abstraction levels to realize the connection between them. For that purpose we might use associations. For example, we might introduce a "supports" association to model the fact that a particular tool supports an activity from the pedagogical level. Thus, we might say that a document viewer "supports" reading activity; a document editor "supports" writing activity; or an annotation tool "supports" commenting activity. In addition to associations between objects and classes from different abstraction levels, the objects and classes from one and the same abstraction level might be associated with each other. For example, the document editor that supports the writing part of the collaborative writing needs to deal with different versions of the documents. Typically, the editor communicates with an external version control tool to manage the document versions. Thus, we can say that the document editor "depends on" the version control tool. Note here, that such associations are of crucial importance for the automatic processing phase where associations might be processed in a number of ways, e.g. to automatically infer the tools that are needed, to highlight dependencies between the tools, or simply to obtain configuration settings that might support a particular learning scenario.

For a better understanding object-oriented domain models are commonly represented in a graphical way. For simplicity we use simple directed labeled graphs to represent object-oriented models of learning scenarios with the following definition of graph semantics. All the objects and classes regardless of abstraction level, as well as all attribute values are represented as graph nodes. The proper abstraction level and the distinction between the objects, classes, and attributes are typically visible from the context of a node. All the structural relations, associations, and attribute names are represented as directed labeled edges between the nodes. Again, the distinction between different edges and their semantics is typically trivial and visible from a wider context of an edge. Figure 3 shows an example of a domain model of QTI that represents quiz, multiple-choice question and true/false question (a specialization of multiple-choice question).

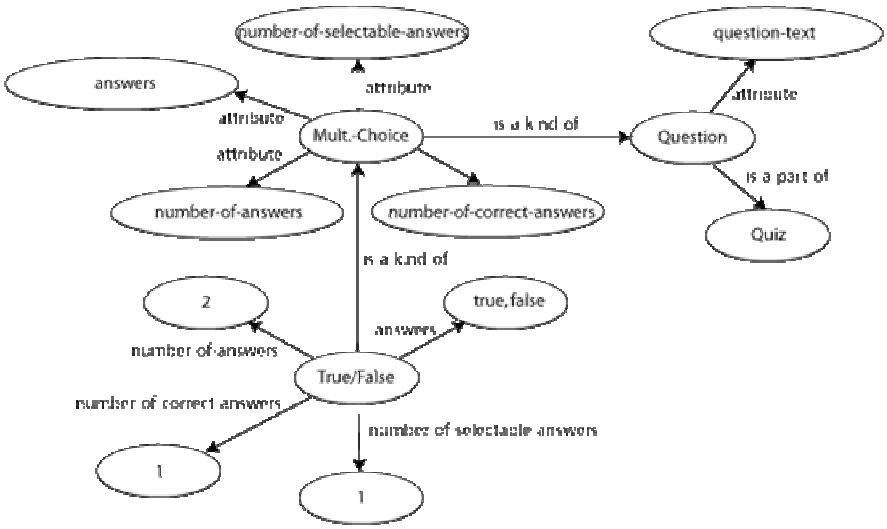

Figure 3: Graphical representation of a part of a QTI domain model

Here a short comparison of the presented meta-model with similar modelling approaches in E-Learning is needed. Recently, an international standard has been developed called IMS Learning Design (IMS-LD). IMS-LD provides a formalized way of describing activity-based learning scenarios and expressing different pedagogical concepts in the form of so-called Units-of-Learning. Units-of-Learning can be exchanged between different E-Learning systems for execution. Additionally, IMS-LD Units-of-Learning might refer to the external learning content (e.g., by linking the content via Web addresses) or might refer to the tools and services available in an E-Learning system [Koper and Burgos, 2005]. However, in the current version of IMS-LD those tools and services are restricted only to four simple services such as e-mail and discussion forum. Also, IMS-LD totally lacks the possibilities to express domain knowledge in a form of general class and composition hierarchies. Basically, an IMS-LD Unit-of-Learning simply defines a sequence of learning activities at the pedagogical level. As such IMS-LD does not include a mechanism to map the learning procedure onto the execution procedure, i.e. onto a system configuration. This means that this part of the learning scenario is basically unsupported by IMS-LD and it is up to the implementers of the IMS-LD specification how such a mapping can be accomplished. This represents a serious drawback of IMS-LD [Leo et al., 2004; Torres et al., 2005].

3.2 Domain Models

Using the presented object-oriented meta-model we can now develop a number of domain models. The domain model development consists of identifying the key classes, their attributes, and relations with other classes in a particular domain. Note

here that such domain models can be at different levels of granularity, i.e. one domain model can be a detailed model of a single class from another model. The possibility to hierarchically structure different abstraction and granularity levels provided by the meta-model makes it highly expressive. However, through the mechanism of encapsulation and information hiding the domain models are kept simple – this allows a possibility to provide a mechanism for automatic processing of such domain models.

3.2.1 Domain Model: QTI

The model of the QTI standard is a rather simple one. Note here that we will not deal with all the features of QTI because that is, obviously, out of the scope of this paper. Rather, a simple conceptual model of the QTI standard will be developed only to illustrate modeling capabilities of the object-oriented meta-model. Thus, in the QTI standard we might identify the following classes:

- Question is a basic unit of the QTI standard. Each question has the text of the question, and might have information on response processing, i.e. feedback for students in the case that the answer was correct, feedback for students in the case that the answer was wrong, or the number of points that students get for a correct answer.
- Quiz (Test) is a collection of a number of question instances.
- Multiple-Choice question is a specialization of the question class that requires from students to choose one or more correct answers from a number of answers. There are several subtypes of the general multiple-choice question such as single correct answer, true/false question, or multiple correct answers. These subtypes provide different default values for some of the attributes of the general concept. Moreover, different types might be identified regarding the format of answers such as textual answers, image answers, image hotspot answers, or mixed answers.
- Fill-in-blank question is another specialization of the question class where students need to type in a correct answer into allocated space in the form of a text. Depending on how the text is interpreted different subtypes of the general concept can be identified. For example, the QTI standard defines the following interpretations: string, integer, and decimal. A special case of the fill-in-blank question is a so-called short essay which defines an alternative way of result processing, namely it allows students to enter free text, and therefore only provides hints for students as a response.
- Drag and drop question where students need to drag objects with mouse to appropriate positions on the screen. Again, there are several subtypes of this question depending of the kind of the objects that can be dragged (textual objects or images). Also, depending on the spatial distribution of objects and spatial direction in which the objects can be dragged a number of subtypes might be identified. For example, sometimes the objects might be dragged only in horizontal or vertical direction making that question a simple objects ordering exercise.

Note here that the QTI standard includes additional question types and a more detailed specification of their attributes, or result processing possibilities. However, with the simple model from above and the capabilities of object-oriented modeling discussed above (i.e. composition, inheritance, and encapsulation) we have been able to create very sophisticated models of QTI quizzes and questions that completely covered scenario requirements in the Ephras project.

3.2.2 Domain Model: WBT-Master

The model of WBT-Master is a rather complex one, since WBT-Master is a fully-fledged E-Learning system that is based on a wide range of pedagogical and pedagogical concepts realized with sophisticated and complex technical infrastructure. For example, the question and test module that is based on the QTI standard is only a single component of WBT-Master. However, as mentioned above inheritance and encapsulation might be used to represent the hierarchical nature of such a system and to hide irrelevant details from a model. Here we will describe only two higher abstraction levels of WBT-Master and will not go into details of different modules.

WBT-Master is based on a sound pedagogical approach, i.e. it is based on collaborative, socio-constructivist, and activity-oriented learning theory. As such it supports students and teachers in a wide range of activities that are centered on a community building process. This pedagogical level is the highest abstraction level of WBT-Master. It includes the following classes:

- User role refers to different pedagogical roles that users might have in a learning scenario, e.g. there are students, tutors, trainers, teachers, or lecturers. Each of these roles is modeled as a specialization of the user role concept.
- Activity is a single task that is executed by a user with a specific role. There exist a wide range of activities, each of them reflecting a particular collaborative learning task. For example, there are activities such as reading, writing, uploading, testing (recollect that if there is a need for detailed modeling of this activity we might use the QTI model from above – similar models might be developed for other activities as well), or communicating. Some of these activities have subtypes, e.g. communicating might be discussion, commenting, providing feedback, chatting, collaborative drawing, and similar.
- Activity pattern is a composite activity that might include a number of other activities in a single new activity. For example, collaborative writing is such a composite activity that has as its components writing, reading, reflecting, and commenting activities.

At the same abstraction level we can identify a so-called learning environment, but we will look onto it separately for comprehension purposes. The learning environment contains the following classes:

- Training (learning) object which is a basic unit of learning material or content. Typically, each activity is associated with one or more training objects, e.g. a reading activity is associated with a document.

- Training course provides a particular learning context by enclosing a number of activities and training objects into a coherent learning entity. In addition, a training course might provide a training curriculum, additional communication features to enhance community building process (e.g. user awareness features, group management features, and similar).

Lastly, let us define classes that are related to the technological infrastructure of the system. Note here that the classes from this abstraction level are typically related with the classes from pedagogical level by means of associations, e.g. by means of the "supports" association. Some of the technological classes include:

- Training object library which is a collection of a number of training objects that supports the creation of training courses.
- Training object editor that supports authoring, editing, and managing of training objects.
- Discussion forum which supports asynchronous moderated or non-moderated discussion and communication.
- Annotation tool which supports commenting and feedback activity.
- Assessment tool which supports grading feedback activity.
- E-Mail which supports asynchronous exchange of messages.
- Instant messaging tool which supports synchronous exchange of messages.
- Chat tool which supports synchronous communication.
- Whiteboard tool which supports synchronous communication enhanced with possibilities to exchange images, vector graphics, or to draw in a synchronous manner.
- Quiz/test module that supports authoring, management, and execution of questions, quizzes and tests.
- Upload tool to support publishing of student results.
- Editor tool that supports writing and editing of documents.
- User management tool that utilizes authentication facilities and user rights management. Typically, all other tool and modules depend on the user management tool.

Note here that the list of technological classes does not raise a claim on completeness – rather it tries to serve only as an illustrative example of a domain model of the technological infrastructure of an E-Learning system.

3.3 Formal binding: XML

To export domain models an XML binding has been developed. The basic XML model is a labeled ordered tree where labels represent node names. Edges are always directed (to preserve the tree order) and do not have labels. Additionally, XML supports a referencing mechanism between nodes, which basically facilitates modeling of arbitrary graphs. In this way directed-labeled graphs might be represented by means of XML documents. An excerpt from an XML document that formally represents hierarchy of multiple-choice question types (see Figure 3) is shown in Listing 1. Note also the representation of attributes.

```
<model>
      <name>QTI</name>
      <class id="1">
            <name>Question</name>
            <attribute key="question_text"
      type="string"/>
            …
      </class>
      <class id="11">
            <is-a-kind-of refid="1"/>
            <name>Multiple Choice</name>
            <attribute key="answer_count" type="int"/>
            …
      </class>
      <class id="111">
            <is-a-kind-of refid="11"/>
            <name>True/False (Text)</name>
            <attribute key="answer_count" default="2"/>
            …
      </class>
      …
</model>
```

Listing 1: XML-based representation of multiple-choice questions

Through the referencing mechanism for XML documents a number of models might be combined. For example, an XML document that represents WBT-Master model might simply refer to the QTI representation to provide a model of its question/test module.

3.4 User Interface

In order to simplify capturing of learning scenarios a user-interface module has been implemented on the top of the defined XML binding. The most important requirement for the user-interface module was the usability, i.e. it should be intuitive and easy-to-use. For that purpose the user-interface module has been developed in the form of a wizard. The wizard leads the users in a step-by-step fashion through the model in question. In addition, it allows them to explore different domain classes in more detail by navigating through the model's hierarchical structure, exploring and learning different model features. For example, the users might retrieve a more detailed description of a particular class, or compare that class with other similar classes or with classes from the same abstraction level. Structural relations ("is-a-kind-of" and "is-a-part-of") serve as the basis for the navigation direction within the wizard, i.e. the users investigate at each moment only a single class but are provided with links to access components of that class or its specializations (see Figure 4). In the cases of multiple inheritances or whenever a class belongs to a number of classes a history stack is utilized to support the users in navigation to the opposite direction.

Additionally, at each navigation step the users might select a particular class to include it in a current learning scenario. If a particular class has attributes they are

first populated with default and inherited values, and in addition the users might be asked to enter remaining attribute values (see Figure 5). Basically, selecting a class for the scenario and populating its attributes with values is equivalent to instantiation in object-oriented sense, i.e. the users create instances of that particular class and define their current state in this way.

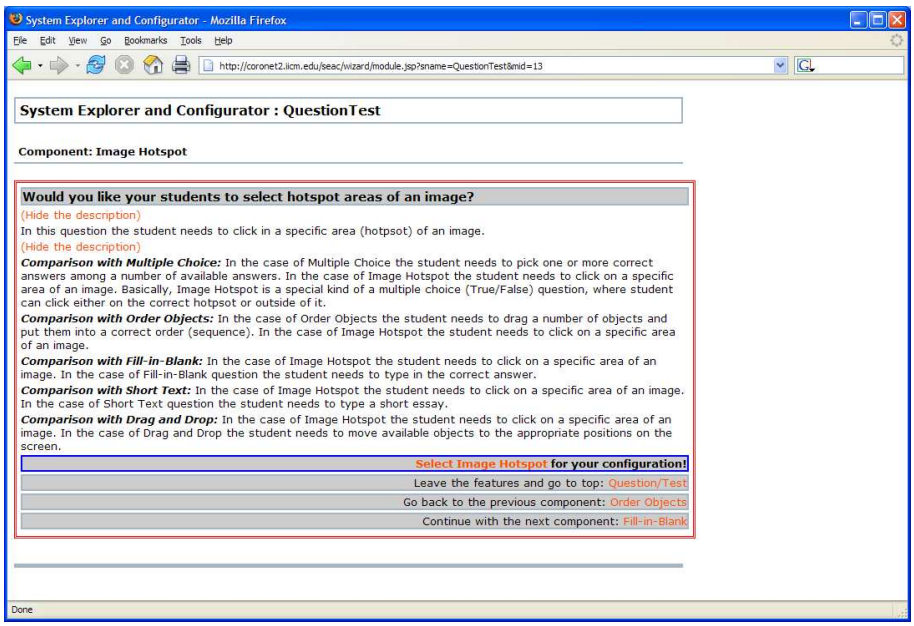

Figure 4: Wizard-like user interface of the QTI model

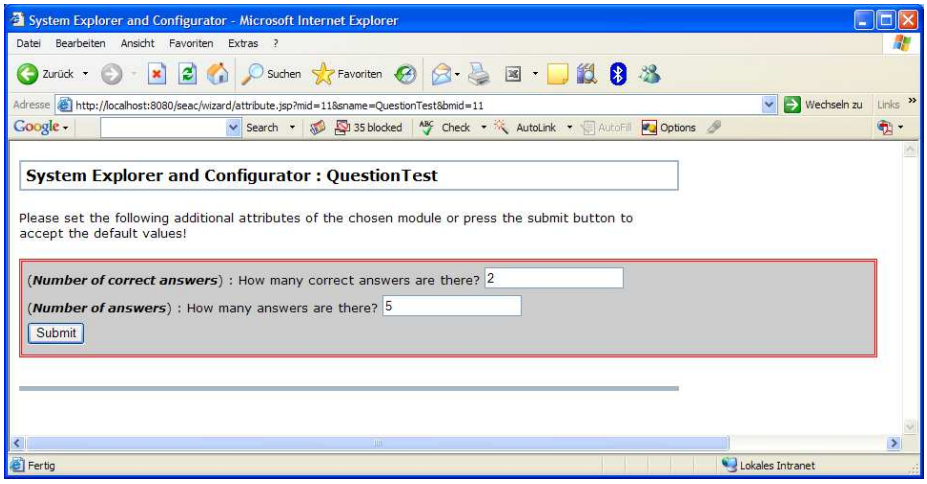

Figure 5: Setting attributes of a multiple-choice question

Figure 6 depicts the relation between a particular domain model and a captured learning scenario. The scenario contains a number of objects that are instances of classes defined in the domain model.

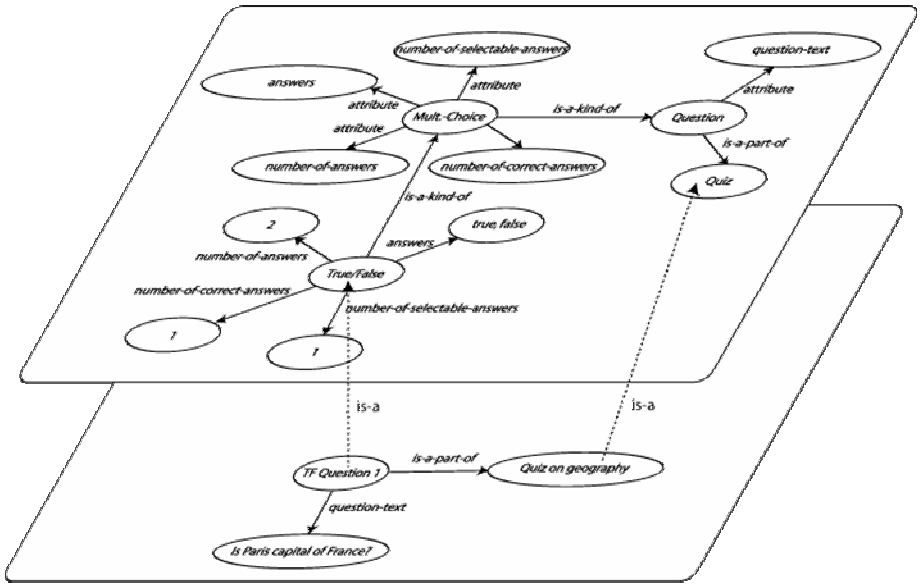

Figure 6: Relation between a domain model and a captured learning scenario

Lastly, after the users finish their exploration session a simple (formal) description of the current learning scenario is obtained. That description lists all of the created class instances together with the values of the associated attributes. The current learning scenario is also represented as an XML document. That document is used as the basis for further automatic processing of the scenario.

3.5 Processing of Scenarios

The formal XML-based representation of a scenario obtained during a user exploration session might be automatically processed to support system configuration. There are several strategies for such an automatic processing – the choice of an appropriate strategy depends on the degree to which a particular system or module can be dynamically configured or customized.

Thus, there are two classes of systems – the first one is a class of systems that are dynamically configurable. For example, a QTI quiz is a simple collection of a number of questions. Since the QTI standard defines an XML binding a QTI quiz might be easily represented as a number of valid XML documents. Obviously, transforming an XML document that captures a scenario onto a number of standard QTI documents is a trivial task that can be accomplished by means of different technologies such as XSLT, a simple transformation script, or a special transformation program. Note here, that the obtained QTI XML documents might be used as a certain kind of templates to

rapidly produce instances of these questions and quizzes. The templates might be loaded into a standard compatible QTI editor to enter the specific question, answers, or feedback. Figure 7 shows such a template that has been used in the Ephras project – the template is a very simple quiz that combines a short essay and a multiple-choice question.

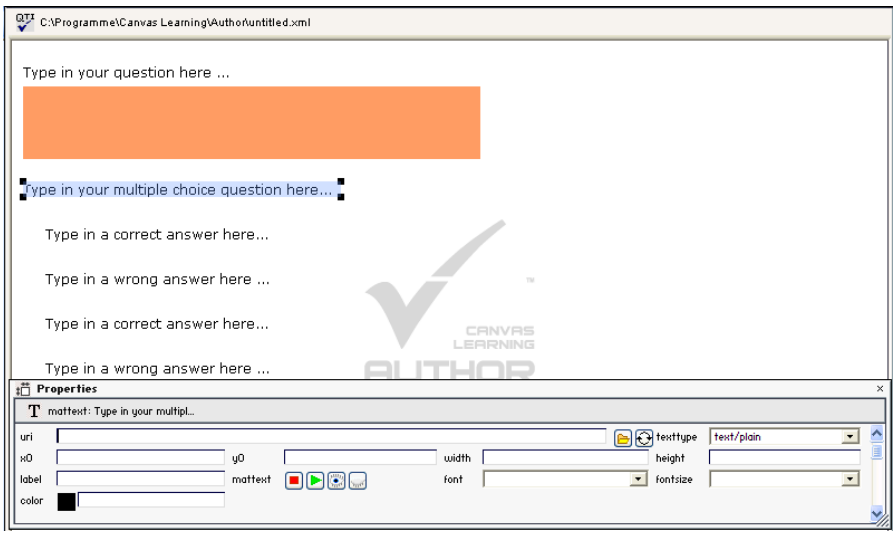

Figure 7: Editing of an automatically created QTI quiz

The second class is a class of systems that exhibit a total or partial lack of possibility for dynamic configuration or customization. For example, some of the features and functionality in WBT-Master can be easily customized in the form of a new training course. However, there are certain features or modules that are preconfigured and their configuration can not be in any way dynamically or automatically adjusted. For instance, a discussion forum in WBT-Master offers a standard possibility to attach a local file to a posting. There is no technical possibility whatsoever to remove that feature from a discussion forum in an automatic way because that feature is hard-coded in the implementation of the discussion forum. Moreover, WBT-Master offers more than 20 predefined and preconfigured training courses which reflect a wide range of common collaborative pedagogical approaches such as project-oriented learning, collaborative writing, or brainstorming. Again, certain modules of these training courses are hard-coded and can not be automatically customized. As a matter of fact, the software architecture of WBT-Master does not provide a possibility for a complete dynamic configuration of a training course, i.e. there is always a need to manually adjust configuration scripts or source code to implement a completely new training course. It is our experience that most of the current E-Learning systems exhibit the same behavior, i.e. it is not possible to configure them dynamically in full. Obviously, in such cases a partial solution should be achieved. How such a partial solution looks like depends on a particular configuration strategy being applied.

Firstly, the system can compare the selections made by the user with the predefined configurations and try to estimate a best-match configuration. The estimation can be implemented in a number of ways. For example, a very simple solution would be to base the estimation on a Boolean model, i.e. whenever the user selects a particular scenario feature the system investigates all predefined configurations and keeps only those where the selected feature is contained. At the next step the system marks the remaining features either as selectable or as non-selectable and reflects this via the user-interface. The selectable features are only those features that are components of one of the remaining configurations. Finally, the users obtain an exact match for their requirements; however, they are restricted in possibilities of what they can select. A slightly better solution might be to use an algebraic or probabilistic model for estimation algorithm. In both of these cases the users are not restricted in what they can select, i.e. they can select any available feature. At the next step, their current selection is modeled as a multidimensional vector (an algebraic model) and compared with vectors representing predefined configurations to obtain an optimal match. Note here, that the obtained match might include features which are not selected at all or might miss certain required features, i.e. it is an approximation of what the users required.

Secondly, the system might simply collect the user requirements without making a particular estimation on the best or optimal match for the collected requirements. Rather, the system might try to produce a prototype configuration which meets the requirements. That prototype can be seen only as a starting point to obtain the proper configuration - at the next step the system developers might work on the prototype to meet the requirements in full.

Thirdly, in some cases even creating a prototype will not be possible. In those cases the collected requirements might be used to configure the system manually from scratch. However, one huge advantage in this case is the fact that the requirements are formally defined as an XML document.

4 First Experiences with Formal Representations of Learning Scenarios

We have applied the presented method for management of learning scenarios at the end of the Ephras project. Additionally, we have started a test phase for the new approach by including the wizard tool into WBT-Master. Here are in short some of the first experiences gained.

4.1 Experiences from Ephras

In the Ephras project the wizard tool has been presented to the teachers at the end of the project. Therefore, the teachers had already some knowledge of what can be expected from the technical realization of their ideas. Nevertheless, they could deepen their knowledge about the technical possibilities by simply using the wizard tool for exploration and reading about different QTI question types. In addition, a possibility to automatically and immediately obtain a quiz and define details of questions that have been chosen for the quiz tremendously improved the understanding of the technical possibilities of the system. This in turn, influenced the way how the teachers

designed their new exercises from the pedagogical point of view. Basically, they focused more on such learning scenarios which are technically feasible and they also introduced new learning scenarios which can be characterized as specific learning scenarios for a computer-aided instruction, i.e., the teachers did not use such learning scenarios in a traditional classroom settings for paper exercises because they lacked the interactive possibilities offered by an E-Learning system.

On the other hand, the system developers got a far better understanding of the pedagogical requirements within the project. By analyzing automatically obtained quizzes, question types, and questions the system developers could identify which question types are the most important for the teachers and, more importantly, the system developers could identify the combinations of the questions which are typically used. For example, one combination that has been very frequently used was a combination of a larger text where a number of sentences might be selected with a single correct selection. In addition, there is a multiple-choice question where a single answer can be selected to replace the selected a sentence from the text. The replacement itself is achieved by means of dragging-and-dropping the selected answer from the multiple-choice question. Technically, this is a very complex combination, that includes a hot-spot area question for selecting a sentence form the text (using hot-spot in this particular case is actually a workaround since QTI does not define a question type "select a text"), a multiple-choice question, and a drag-and-drop question. Note that hot-spot area and drag-and-drop questions include also a spatial component, i.e., the developer needs to define positions on the screen where the answers are located or might be dropped with the mouse.

To implement and configure such combinations the developers introduced so-called templates, i.e. generic implementations of the desired behaviour that can be used to create instances by configuring it. The configuration process typically involves setting of a couple of parameters. However, the problem was how to recognize that the teacher wants to have a particular template. The solution for the problem is very simple. The developers simply included the created templates into the domain model for QTI as special classes that combined a number of questions into a single entity. In this way the teachers could simply explore a template as any other class from the model and see its components. If the template fits into their scenario requirements the teachers simply include it in the scenario. Note how the meta-model through its extensibility (supported in this case by means of a composition mechanism) greatly facilitated this problem.

The last difficulty that the system developers have been confronted with is not directly related to the presented approach but to the QTI standard and the way the presentation of questions is defined in QTI. Basically, QTI XML files mix together the content and its presentation. For each question apart from its actual content the developer needs to define its presentation, e.g. the coordinates on the screen where the question will be presented. This is a serious design flaw since it violates a well-known principle in software development called separation of concerns [Dijkstra, 1982]. Consequently, whenever the developer modifies the content of a question its presentation has to be modified as well because the spatial relations within the question need to be updated. Especially, creating template instances can be very tedious because of this difficulty. Referring to the example from above suppose that the text (from which sentences might be selected) is modified. This means that the

developer needs to update the positions of hot-spot areas as well as position of areas where replacement sentences might be dropped. The only way to achieve this when using QTI is to update the template instance manually either by using a QTI editor or by editing QTI XML files with an XML editor.

4.2 Experiences with WBT-Master

WBT-Master belongs to the class of systems that cannot be fully dynamically configured, i.e. the system offers about 20 predefined configurations from which a single configuration might be selected. Additionally, the system developers can introduce a new configuration to fit the requirements of a particular learning scenario by manually creating a new training course.

To select a particular configuration, the users (teachers of Civil Engineering courses at the University of Technology Graz) operated the wizard tool to explore the WBT-Master domain model and select the features which fit into their learning scenario. The system then offered an optimal preconfigured training course that was the best fit for the selected pedagogical features. Here two configuration strategies have been applied.

The first strategy was based on a Boolean model, i.e. whenever the users select a feature all other features not related with the selected one in any of the existing configurations cannot be selected anymore. However, the users can still explore all of the existing features and learn in that way more about the system. This approach has been successful with novel users of the system because at the end of a learning scenario capturing session the training course that was configured by the wizard tool was an exact match of what the users selected.

For more experienced users who were already familiar with WBT-Master this approach was too restrictive. Therefore, for experienced users we based the configuration strategy on an algebraic model (a vector space model). Here, the users could select any feature to include it in their learning scenarios. The selected features were then modelled as a multidimensional vector and then compared with predefined configurations (also modelled as multidimensional vectors) to find an optimal match. The success of this configuration strategy depends strongly on the underlying vector space model. In particular, it depends on the dimensions of the vector space, as well on how these dimensions are weighted in the model. For WBT-Master we included for example communication, tutoring, self-initiated learning, testing, content creation, or student content creation as dimensions in the vector space model. At the next step each feature has been modelled as a vector in this vector space, and a particular learning scenario or a predefined configuration is simply a sum of vectors of all selected features. Obviously, by selecting appropriate dimension weights it is possible to emphasize that a particular dimension is more important than the other one. In the case of WBT-Master the communication dimension and student content creation dimension have been weighted the most.

The approach based on non-restrictive selection of pedagogical features has another important property. Basically, we have used this approach to collect the data about desired and needed learning scenarios. At the next step we have analyzed that data and tried to identify certain usage patterns of pedagogical features, e.g. how pedagogical features are typically combined. This information has been used to develop a number of new training courses that reflected the identified patterns. These

training courses have been included in WBT-Master, as well as in its domain model for the wizard tool. Thus, the system evolution has been facilitated in this way.

5 Conclusions and Further Work

This paper argued that modern E-Learning systems which support pedagogical aspects of E-Learning are inevitably technically complex. Also, there is a gap between the pedagogical and technical aspects in such systems. Typically, this situation leads to suboptimal usage of the systems. To bridge that gap a huge effort on both sides, i.e. the teacher side as well as the system developer side is needed. In particular, in order to support pedagogically sophisticated learning scenarios an iterative system configuration and management process is needed. The paper presented a possible solution for this problem based on formal representation of learning scenarios by modelling the pedagogical as well as technical domain in question. The first results with the presented solution were positive, especially in such E-Learning systems where a complete dynamic configuration is possible. In a system where that is not the case the presented solution tries to estimate an optimal configuration. Additionally, it supports the system evolution through the analysis of the user's needs.

Although the results of the presented approach are encouraging we see it only as a first step in automatic management of learning scenarios. Currently, this approach does not take into the account the dynamics of a learning scenario. Thus, the users select the pedagogical features which they would like to include in a learning scenario but they cannot impose a structure on top of these features. For example, for collaborative writing scenario a user would include activities such as writing, reading, reflecting, commenting, and discussion. The system would offer a particular configuration that includes tools to support each of these activities. All of these tools would be presented to the students in a single user-interface and will be present at all times. However, there is a certain temporal structure which can be imposed on the top of these activities, i.e. firstly, the students need to read a particular document; secondly, the students need to write their own document; thirdly, the teacher provides comments; lastly, students are involved in reflecting activity. In parallel, the discussion activity is carried out. Thus, there is a certain (process-oriented) execution sequence within this particular learning scenario and that execution sequence should be also supported by the system. For example, during the writing activity the students should have at their disposal only a text editor to write their contributions and a discussion forum to discuss all current issues with the teacher and their peers. At the next step during the commenting activity the students can only read the teacher's comments and do not have the text editor at their disposal anymore. In this way the learning process within a learning scenario might be facilitated. Thus, our future work would include an extension of the presented approach to include management of the process-oriented aspects of learning scenarios.

References

[Avgerou, 1987] Avgerou, C. The Applicability of Software Engineering to Information Systems Development. *Information & Management,* 13, pp. 135-142, 1987.

[Bolchini and Paolini, 2004] Bolchini, D., Paolini, P. Goal-driven requirements analysis for hypermedia-intensive Web applications, Requirements Engineering, Volume 9, Issue 2, May 2004, pp. 85-103, 2004.

[Booch 1993] Booch, G. Object-oriented analysis and design with applications (2nd ed.), Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, 1993.

[Brooks 1987] Brooks, F.P. No Silver Bullet Essence and Accidents of Software Engineering, *Computer*, Volume 20, No. 4, pp. 10- 19, 1987.

[Clarke and Wing, 1996] Clarke, E. M., and Wing, J. M. Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*, Volume 28, No. 4, pp. 626-643, 1996.

[Dijkstra, 1982] Dijkstra, E. W. On the role of scientific thought, in *Selected Writings on Computing: A Personal Perspective*, Springer-Verlag, pp. 60-- 66. 1982.

[Dreher et al., 2004] Dreher, H., Scerbakov, N., Helic, D. Thematic Driven Learning, *Proceedings of E-Learn 2004*, pp. 2594-2600, AACE, Charlottesville, USA, 2004.

[Ebner and Holzinger, 2002] Ebner, M., and Holzinger, A. E-Learning in Civil Engineering: The experience applied to a lecture course in Structural Concrete. *Scientific Journal of Applied Information Technology (JAPIT)*, pp. 1-9, 2002.

[Ebner and Holzinger, 2003] Ebner, M., and Holzinger, A. Instructional Use of Engineering Visualization: Interaction Design in e-Learning for Civil Engineering. In Jacko, J. and Stephanidis, C. eds. *Human-Computer Interaction Theory*, 2003.

[Ebner et al., 2005] Ebner, M., Scerbakov, N., Maurer, H. New Features for eLearning in Higher Education for Civil Engineering, Proceedings of E-Learn 2005, pp. 635-642, AACE, Charlottesville, USA, 2005.

[German, 2000] German, D. M. Hadez, a framework for the specification and verification of hypermedia applications. PhD Thesis, University of Waterloo, 2000.

[Helic et al., 2005] Helic, D., Krottmaier, H., Maurer, H., Scerbakov, N. Enabling Project-Based Learning in WBT Systems, *International Journal on E-Learning (IJEL)*, Vol. 4, Num. 4, pp.445-461, 2005.

[Helic et al., 2004] Helic, D., Maurer, H., Scerbakov, N. Knowledge Transfer Processes in a Modern WBT System, *Journal of Network and Computer Applications*, Vol. 27, Num. 3, pp.163-190, 2004.

[Helic, 2006] Helic, D. Template-based Approach to Development of Interactive Tests with IMS Question and Test Interoperability, *Proceedings of ED-MEDIA 2006*, pp. 2075-2081, AACE, Charlottesville, USA, 2006.

[Helic and Durco, 2005] Helic, D., Durco, P. Aspects of an XML-Based Phraseology Database Application, *Proceedings of the Third International Seminar on Computer Treatment of Slavic and East European Languages*, SLOVKO 2005, pages 99-108, VEDA, Bratislava, Slovakia, 2005.

[Helic et al., 2005a] Helic, D., Maurer, H., Scerbakov, N. A Didactics Aware Approach to Knowledge Transfer in Web-based Education, In Claude Ghaoui, Mitu Jain, Vivek Bannore

(Editors), *Studies in Fuzziness and Soft Computing*, Volume 178/2005, Chapter 9, pp 233-260, Springer-Verlag GmbH, 2005.

[Hirumi, 2002] Hirumi, A. The Design and Sequencing of E-Learning interactions: a Grounded Approach. *International Journal on E-Learning*, 1(1), pp. 19–27, Norfolk, VA: AACE, 2002.

[Hobbs, 2002] Hobbs, D. L. A Constructivist Approach to Web Course Design, a Review of the Literature. *International Journal on E-Learning*, 1(2), pp. 60–65, Norfolk, VA: AACE, 2002.

[Holzinger and Ebner, 2003] Holzinger, A., Ebner, M. Interaction and Usability of Simulations & Animations: A case study of the Flash Technology. *Proceedings of Interact 2003*, pp. 777-780, 2003.

[Hong and Lingzi, 2000] Hong, Z., Lingzi, J. Scenario Analysis in an Automated Tool for Requirements Engineering, *Requirements Engineering*, Volume 5, Issue 1, Jul 2000, pp. 2–22, 2000.

[IMS QTI, 2005] IMS Global Learning Consortium Question & Test Interoperability Specification (QTI), http://www.imsglobal.org/question/

[Jarke et al., 1998] Jarke, M., Bui, X. T., and Carroll, J.M. Scenario Management: An Interdisciplinary Approach. *Requirements Engineering*, 3, 3/4, pp. 155-173, 1998.

[King and Puntambekar, 2003] King, F., and Puntambekar, S. Asynchronously Conducted Project-based Learning: Partners with Technology. *International Journal on E-Learning*, 2(2), pp. 46–54, Norfolk, VA: AACE, 2003.

[Koper and Burgos, 2005] Koper R. and Burgos D. Developing advanced units of learning using IMS Learning Design level B. *International Journal on Advanced Technology for Learning*, Vol. 2, Num. 4., pp.252-259, 2005.

[Land and Hirschheim, 1983] Land, F. F., Hirschheim, R. A. Participative Systems Design: Rationale, Tools and Techniques. *Journal of Applied Systems Analysis,* 10, pp. 91–107, 1983.

[Leasure et al., 2000] Leasure, A., Davis, L., and Thievon, S. Comparison of Student Outcomes and Preferences in a Traditional vs. World Wide Web-based Baccalaureate Nursing Research Course. *The Journal of Nursing Education*, 39(4), pp. 149–154, 2000.

[Leo et al., 2004] Leo D.H., Perez J.I.A., Dimitriadis Y.A. IMS learning design support for the formalization of collaborative learning patterns, *Proceedings. IEEE International Conference on Advanced Learning Technologies, 2004.* 30 Aug.-1 Sept. 2004. pp.350 – 354, 2004.

[Meyer, 1985] Meyer, B. On Formalism in Specifications. *IEEE Software*, 2, 1 (January), pp. 6-26, 1985.

[Milligan, 2003] Milligan C. Question and Test Interoperability (QTI): Extending the specification for Mathematics and Numerical Disciplines*, Maths CAA Series*, The Maths, Stats & OR Network, University of Birmingham, UK, 2003

[Mioduser et al., 2000] Mioduser, D., Nachmias, R., Lahav, O., and Oren, A. Web-based learning Environments: Current Pedagogical and Technological State. *Journal of Research on Technology in Education*, 33(1), 2000.

[Motschnig-Pitrik and Holzinger, 2002] Motschnig-Pitrik, R., Holzinger, A. Student Centered Teaching Meets New Media: Concept and Case Study*, IEEE Journal of Educational Technology & Society,* Vol. 5, pp. 160-17*,* 2002.

[Oliver et al., 2002] Oliver, R., Harper, B., Reeves, T., Strijker, A., and Westhuizen, D. Learning Management Systems: One Size Fits All? *Proceedings of World Conference on*

Educational Multimedia, Hypermedia and Telecommunications, pp. 1498– 1499, Norfolk, VA: AACE, 2002.

[OMG, 2003] Object Management Group: *Model Driven Architecture,* http://www.omg.org/mda

[Overmyer, 2000] Overmyer, S. P. What's Different about Requirements Engineering for Web Sites? *Requirements Engineering*, Volume 5, Issue 1, Jul 2000, pp. 62-65, 2000.

[Pfahl et al., 2004] Pfahl D., Trapp S., Helic D. A Methodology-Driven Software Infrastructure for Work-Based Learning, Michael Kelleher, Andrew Haldane, Eelco Kruizinga (Editors), *Researching Technology for Tomorrow's Learning: Insights from the European Community*, Chapter 3, pp.85-95, CIBIT Consultants|Educators, 2004.

[Smythe and Roberts, 2000] Smythe, C., and Roberts, P. An Overview of the IMS Question & Test Interoperability Specification, *Computer Aided Assessment (CAA'2000)*, Leicestershire, UK, 2000.

[Torres et al., 2005] Torres J., Dodero J.M., Aedo I., Zarraonandia T. An architectural framework for composition and execution of complex learning processes, *Fifth IEEE International Conference on Advanced Learning Technologies, 2005*. ICALT 2005. 5-8 July 2005. pp.143 – 147, 2005.