Complex Course Generation Adapted to Pedagogical Scenarios and its Evaluation

Carsten Ullrich
Dept. of Computer Science and Engineering, Shanghai Jiao Tong University
1954 Huashan Road, 200030 Shanghai, China
ullrich_c@sjtu.edu.cn

Erica Melis
DFKI GmbH
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
melis@dfki.de

ABSTRACT

A course(ware) generator (CG) assembles a sequence of educational resources that support a student in achieving his learning goals. CG offers a middle way between pre-authored "one-size-fits-all" courseware and individual look-up of learning objects. Existing course generators however, incorporate only limited CG knowledge. They only handle a single, limited type of course and cannot handle scenarios. In this paper, we present a course generator that implements a previously unrealized amount of pedagogical knowledge. We illustrate the expressivity of this CG knowledge by describing six different scenarios. An additional novel feature is that the courses it generates are structured in sections and subsections which makes orientation and navigation easier for students. We also present the results of Europe-wide formative and summative evaluation. The evaluation investigated the students' view on CG in general and for each scenario in particular. The data show that the realized adaptivity is appreciated by the learners and that the learner-driven usage of the course generator helps learners to find they own way of learning and makes them feel being respected, treated as adults.

Keywords

adaptivity, pedagogical knowledge, courseware, learning objects

Introduction

A course(ware) generator assembles a sequence of educational resources that support a student in achieving his learning goals. The selection of the learning resources takes information about the learner into account, for instance his competencies and preferences. Course generation (CG) offers a middle way between pre-authored "one-size-fits-all" courseware and individual look-up of learning objects (Brusilovsky & Vassileva, 2003). Ideally, a generated course specifically addresses the user's current learning purpose and contains all learning opportunities that are required to achieve the purpose. In the following, we call this purpose or type of course *scenario*. For instance, a student who wants to rehearse the derivation procedures (scenario *rehearse* for the target concept "derivation procedures") needs a "course" that is different from one for a student who wants to train modeling of problems for derivation procedures (scenario *train competency Model*, same target concept as above).Thanks to pedagogical experience and theories, human teachers are able to adapt their teaching to the individual students and their needs. Part of this experience and knowledge can be represented by knowledge about which learning resource is most adequate for the individual learner at a specific point in time and how the concepts should be ordered, i.e., how difficult exercises and examples should be, which concepts have to be elaborated in the first place, etc.

However, today's state of the art course generators incorporate only limited pedagogical knowledge, which is particularly remarkable considering that CG has long been a research topic and how advanced the early approaches have been in this regard.. The "Generic Tutoring Environment" (GTE) (Van Marcke, 1992) and the "Dynamic Courseware Generator" (DCG) (Vassileva & Deters, 1998) are course generators based on the explicit modeling of knowledge. Despite the time that has passed since their development, no other CG system possesses a comparable large instructional knowledge base up to now, except of Paigos, the course generator presented in this article. However, they have no notion of scenarios, only a limited selection of exercises and obviously do not integrate today's state of the art Web technology that enables reuse of learning materials from different systems and repositories.

Later approaches (Specht & Oppermann, 1998, Specht et al., 2001, Conlan et al., 2003, Keenoy et al., 2003, Méndez et al., 2004) put less focus on pedagogical knowledge or do not provide sufficient descriptions to assess it. None of these systems generates structured courses as Paigos does. In Paigos, the hierarchical knowledge encoded in the pedagogical methods is reflected in the different sections and sub-sections, thereby providing a means for the student to understand the structure of a course.

(Karampiperis & Sampson, 2005) suggest using statistical methods for determining the best path through the educational resources. A major drawback is that the educational resources are rated by an instructional designer with respect to a specific learner. Additionally, the rating does not take into account that the same learner can have different learning needs regarding the same concepts.

(Sicilia et al., 2006) present the idea to use HTN-planning to generate IMS LD instances. Their description is on a very abstract level and sketches some basic operators and methods. By using two HTN methods as an example, they hint at how different pedagogical approaches ("content-oriented" vs. "socio-cultural") might be implemented. Their work provides evidence that CG techniques can be used for the generation of learning designs; however they do not provide any detailed formalization. Their work is not implemented.

To summarize, existing systems only generate a single, limited type of course and cannot handle complex scenarios. The generated courses consist of all resources that the student still needs to interact with in order to reach the target concepts, starting with the currently known concepts. Moreover, the systems only generate flat sequences of resources which lack the structure of manually authored courses (section, subsections, introductory texts, etc). It is also difficult to plug in third party repositories, and thus these systems can generate courses only for their own database of learning materials.

In addition, in-depth evaluations of course generators are rare. The recent evaluations of CG has either been done on a conceptual level without having been implemented sufficiently for evaluations (Conlan et al., 2003, Keenoy et al., 2003, Méndez et al., 2004, Sicilia et al., 2006) or the evaluations were based on simulations (Karampiperis & Sampson, 2005).

In this article, we describe Paigos, a course generator used in the Web-based learning environment ActiveMath (Melis et al., 2006). We start in Section 2 by explaining how Paigos implements significant amount of pedagogical knowledge that enables it to generate courses for different scenarios. In Section 3 we describe the European-wide evaluation of Paigos. We conclude the article with a summary and an outlook on future work (Section 4).

Knowledge-Based Adaptation of Courses

Example

We start with an example that illustrates how a student uses Paigos and how Paigos generates courses.

Guided by a wizard, student Anton first selects a scenario from a list. Since he wants to learn new topics, he selects the scenario *discover* (in contrast to *rehearse*, for instance). In a second step, he selects the domain concepts he wishes to learn about, for example, the differential quotient, derivative function and sum rule. Finally he starts the CG process. From Paigos viewpoint, the student has specified the task "*discover differential quotient, …*". The implemented pedagogical knowledge tells Paigos to decompose this task into subtasks, according to the structure of scenario *discover*: for instance, each concept should be introduced by a section containing one or two examples or a real world problem (subtask "*introduce differential quotient*"), followed by a presentation of the prerequisite concepts the student needs to know (subtask "*show_prereq differential quotient*"). Then, a section about the concept itself is inserted, followed by a section of practice opportunities, and a section that illustrates how the concept is connected to related concepts. Continuing this process, Paigos generates a course with sections that reflect this structure. Then, each of the sections is refined until all sections are filled with educational resources, the course is generated and is presented to the user. On an average computer, this process happens almost real-time in less than a second.

Figure 1 contains a screenshot of Anton's course. The page displayed at the right hand side of the screenshot is the second page of the course. It contains the first items of the prerequisites page: the text generated to describe the purpose of the section and the prerequisite concepts (only one is visible in the screenshot). The sections displayed in the table of contents (left) vary in the pages they contain. For instance, the first section does not contain an introduction page. The reason is that no appropriate elements could be found for this page and therefore, the page was skipped.

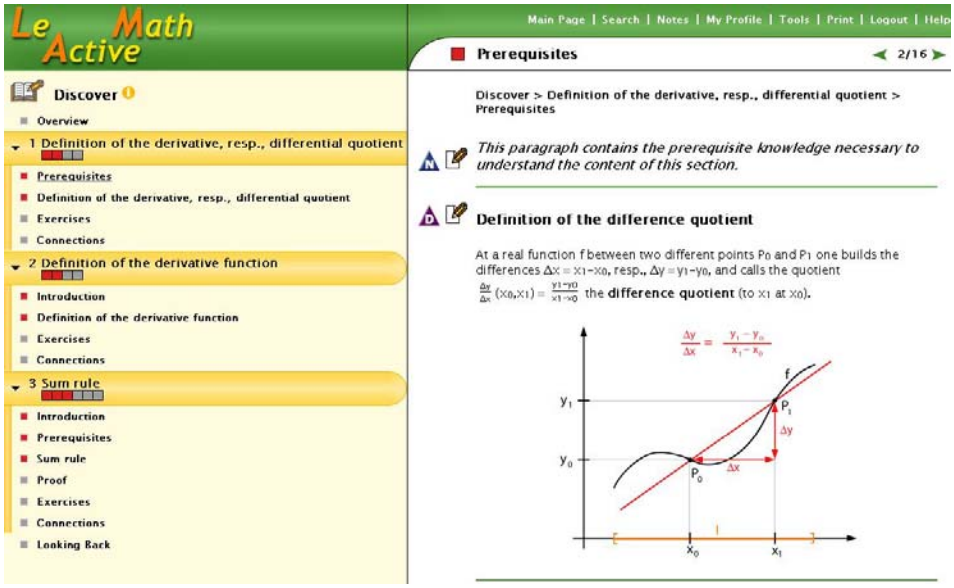

Figure 1. An example of a course of type discover

The Knowledge-Based Technology

In this section, we briefly present basic terminology and technology, as well as the AI framework used in Paigos.

Wiley, (2000) defines learning objects as "any digital resource that can be reused to support learning". In the context of the work presented in this paper, we also need to address how to locate a learning object and its granularity. We therefore define the concept educational resource: an educational resource is a digital resource that can be used and reused for learning, that has a URI (Uniform Resource Identifier) and that is atomic and self-contained – such as an example, definition, a blog entry, etc. The small granularity is essential for a flexible reuse. Educational resources are stored in repositories. Paigos employs a mediator architecture and Semantic Web technology to enable a flexible integration of repositories (Kärger et al., 2006). To connect Paigos to a repository, one needs to specify an ontology T that describes the metadata of the target repository and a mapping between Paigo's ontology of instructional objects (Ullrich, 2005) and T. In general, authoring resources that are usable with Paigos is not more time consuming than authoring LOM/SCORM compliant learning objects. Furthermore, the mediator approach allows reuse of existing resources and in particular thanks to the expressibility of Paigos in pedagogical different ways.

A learning-support tool is any application that trains specific skills and that can be integrated into the learning process automatically. For instance, some Intelligent Tutoring System can serve as a learning-support tool.

Information about the learner is stored in a student model. This can include information about the learner's mastery level, his competencies, which educational resource he has seen, etc.

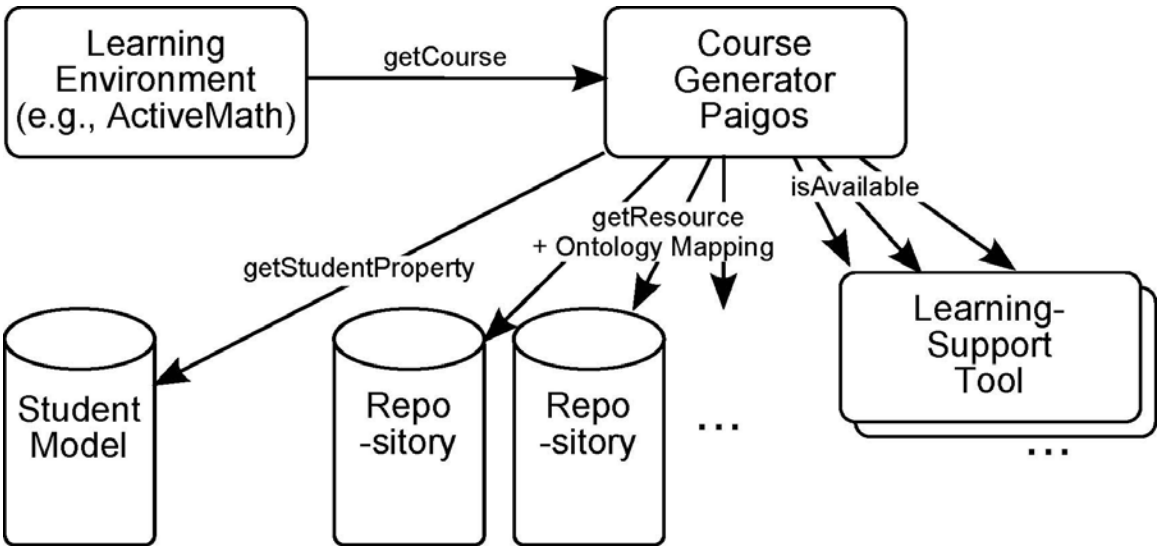

Figure 2. An overview on Paigos

Figure 2 gives an overview on the general setup of Paigos, the course generator described in this article. Paigos is invoked by a learning environment, which needs to specify the user's name, the scenario to be used for the course and the goal concepts of the course. In the example above, the scenario is *discover* and the goal concepts are the differential quotient, derivative function and sum rule. While Paigos is generating a course, it retrieves information from the student model about the learner and from registered repositories about the educational resources that are available. Depending on the pedagogical knowledge, Paigos might also integrate learning-supporting tools in the course, but only if these tools are available for the given learning environment. Tools can be integrated in varying degrees, in the simplest way by presenting a link to the tools. Paigos can also configure the mode in which tools start (e.g., an explorative vs. guided mode for a concept mapping tool), which requires an API by the tool. The precise interaction with the tools, how they are started, the API to call, etc., is outside the pedagogical expertise of Paigos and thus not encoded in the methods. Similarly, feedback from the tools to the LM is handled outside of Paigos.

Paigos uses the Hierarchical Task Network planning (HTN, Nau et al., 2003) as a means to implement and employ the pedagogical knowledge of how to generate a course. In HTN-planning, the goal of the planner is to achieve a list of tasks, where each task is a symbolic representation of an activity to be performed. The planner formulates a plan by using methods to decompose the top tasks into smaller subtasks until primitive tasks are reached that can be carried out directly using operators.

We identified and implemented several basic building blocks (methods and operators) for CG. These basic blocks are pedagogically neutral and independent of the domain to be

taught. They encompass the following actions: 1) inserting resources in a course; 2) structuring courses by opening and closing sections and by inserting dynamically generated texts that inform the learner about the purpose of sections; 3) interacting with knowledge sources, i.e., querying repositories and the student model; 4) combining information from knowledge sources, e.g., removing all known educational resources that are not needed; 5) inserting links to learning-support services; 6) dynamic subtask expansion that stop plan detailing above the actual resource selection. With these building blocks, complete scenarios as described in the following sections can be generated. For the complete formalization as an HTN-planning problem, see Ullrich & Melis, 2009.

Dynamic texts and dynamic subtask expansion (points 2 and 6 above) are novel features for CG. Dynamic text generation uses information provided by the course generator to generate texts that ensure smooth transitions between the individual resources in the course. Dynamic subtask expansion stops CG at a level that specifies what kind of educational resources should be selected but does not specify which ones. The specific resources are selected at the time when the learner wants to use them. This allows generating a complete table of contents of the course while using up-to-date information for the selection of individual resources. Dynamic subtasks can also be used for manually composed courses where parts of the course are predefined and others dynamically computed.

Adaptation to Pedagogical Scenarios

The CG scenarios implemented in Paigos are developed according to moderate constructivist principles and based on competencies. Competency-based pedagogy argues that literacy, e.g., in mathematics, requires mastering different competencies such as performing calculations and being able to argue mathematically (Niss, 2002). The competency level of a student with respect to a specific concept specifies the complexity of exercises the student is able to solve. It ranges from Level I (computation at an elementary level) to level IV (complex processing such as modeling and argumentation).

We identified six major learning scenarios. The scenarios are *discover* new content, *rehearse* weak points, establish *connections* between concepts, *train intensively*, *train competencies* and *exam simulation*. For each scenario, we formalized the corresponding CG knowledge. This knowledge determines the basic structure of a course. For instance, it determines that an example should follow a definition. The courses are adaptive, that is, different resources can be selected by taking information from the student model into account, such the student's success in solving previous exercises. The underlying pedagogical knowledge was formalized in close collaboration with pedagogical experts and based on the literature (Novak & Gowin, 1984, Pintrich, 1999, Reinmann-Rothmeier & Mandl, 2001, Zech, 2002, Tergan, 2002, Prenzel et al., 2004).

Scenario Discover

In the scenario *discover* Paigos generates courses that contain those educational resources that support the learner in reaching an in-depth understanding of the concepts specified by the learner (see Figure 1 for an example). The course includes the prerequisite concepts that are unknown to the learner. It also provides the learner with several opportunities to use learning-support services.

The basic structure of the scenario is adopted from Zech, (2002) and consists of several stages that typically occur in learning a new concept. For each stage, the course contains a corresponding section:

1. Description. The course starts with a description of its aim and structure. Then, for each goal concept selected by the learner, the following sections are created:
2. Introduction. This section motivates the usefulness of the concept using adequate resources such as examples and introduction texts. It also contains the unknown prerequisites concepts.
3. Develop. This section presents the concept and illustrates how it can be applied.
4. Proof. For some concepts, such as mathematical theorems, proofs or other evidence supporting the concepts is presented.
5. Practice. This section provides opportunities to train the concept.
6. Connect. This section illustrates the connections between the current concept and related concepts.
7. Reflection. Each course closes with a reflection section, which provides the learner with opportunity to reflect on what he has learned in the course.

```
1  (:method (discover ?concepts) ; task achieved by the method
2           () ; empty precondition
3           (  ; subtasks of method:
4            (!startSection Discover ?concepts)
5            (descriptionScenarioSection ?concepts)
6            (learnConceptsDiscover ?concepts)
7            (reflect ?concepts)
8            (!endSection)))
9
10  (:method (learnConceptDiscover ?c) ; task achieved by the method
11           () ; empty precondition
12           (  ; subtasks of method:
13            (!startSection Title (?c))
14            (introduceWithPrereqSection ?c)
15            (developConcept ?c)
16            (proveSection ?c)
17            (practiceSection ?c)
18            (showConnectionsSection ?c)
19            (!endSection)))
```

*Figure 3.*Top-level decomposition in the scenario discover

The two methods shown in Figure 3 illustrate what the formalized CG knowledge looks like. In the figure, terms starting with "?" denote variables; a semicolon starts a comment and the following text until the end of the line is ignored. We will not go into every

detail, but convey the general principles underlying the formalization. The methods start the generation of a course for the scenario *discover*. The upper method decomposes a given task (`discover concepts`) (line 1) into five sub-tasks (lines 4–8). Please note that task refers to HTN-tasks, to be "achieved" by the HTN-planner. These are different from tasks that students need to achieve. Since the method has no precondition (the empty list in line 2), the method can always be applied. When a method is applied to a task, then the variables are instantiated with the given values, here, `?concepts` will be instantiated with the list of concepts the course is about. The subtasks of the method do the following: first, a new section is started, in this case the course itself (line 4). Then, a description about the course's aims and structure is inserted (line 5). The third subtask triggers a method that recursively inserts the task (`learnConceptDiscover g`) for each identifier `g` in the list of identifiers bound to `?concepts` (the method that decomposes this task is not shown here). The last two subtasks insert the reflection section and close the course. For each concept `g`, a task (`learnConceptDiscover g`) is created.

The bottom method in Figure 3 decomposes the task into subtasks which closely resemble the structure of the scenario as described above. An introduction to a concept in the scenario *discover* is assembled as follows. First, a new section is started containing a text that explains the purpose of the section. Then, three tasks try to insert several resources: a resource that is of the type *introduction* to the concept, a resource that provides a real-world-problem involving the concept, and an example that illustrates its application. An additional section contains the prerequisite concepts that the learner needs to see. In the scenario *discover*, all prerequisite concepts that are unknown to the learner are presented on a single page. Thus, the student can easily distinguish between the target concepts and the prerequisites.

The section develop presents the concept itself, together with auxiliaries that help the learner to understand it. In case the learner exhibits a high competency level only a single example illustrates the concept. Otherwise, first a text explaining the concept is inserted and followed by several examples that aim at fostering the learner's understanding of the concept.

Proofs and other evidences play an important role in mathematics and being able to prove is one aspect of the competency argue. In case the learner has a high competency *argue*, then she is able to find proofs or establish evidence on her own. In this case, the methods do not insert the proofs themselves but an exercise for the competency *argue*. Otherwise, resources of type *proof* are inserted. In case several appropriate evidences exist, the evidences are inserted ordered by increasing abstractness. The abstractness is determined by the media type of a learning resource: visual evidence (e.g., an illustration) is presented first, followed by verbal, numeric, and symbolic evidence, and finally evidence that is not annotated with a media type.

The methods in the section *practice* insert a number of exercises that provide the learner with opportunities to develop her own understanding of the concept, from a variety of perspectives. The selection of exercises and examples is highly dependent of the

individual student model. It takes the current competency level of the learner and competencies into account. In total, about 60 methods encode the knowledge of exercise and example selection (including specialized rules that e.g., allow selecting resources for specific competencies). In this scenario, the selected resources should cover all competencies (as far as corresponding resources exist).

The section *connect* illustrates the connections between the current concept and related concepts of type *theorem*. If a concept mapping tool is available, then it is used for displaying the connections. Otherwise, the related theorems themselves are inserted in the course, together with explanations and evidence

Each course generated for the scenario *discover* closes with a reflection step, which provides the learner with opportunity to reflect on what he has learned in the course.

Other Scenarios

To illustrate the expressivity of Paigos, we now describe the remaining five scenarios implemented in the LeActiveMath project.

Scenario Rehearse

Courses of the type *rehearse* are designed for learners who are already acquainted with the target concepts but do not yet master them completely. Such a course provides several opportunities to examine and practice applications of the concepts and illustrates the connections between concepts. The structure is as follows:

1. Description. The course starts with a description of its aim and structure. Then, for each goal concept selected by the learner, the following sections are created.
2. Rehearsing. This section presents the concept of the section.
3. Illustrate. This section presents example applications of the concept.
4. Connect. This section illustrates the connections between the current concept and related concepts.
5. Practice. This section provides opportunities to train the concept.
6. Illustrate–2. This section contains additional examples.
7. Practice–2. This section contains additional exercises.

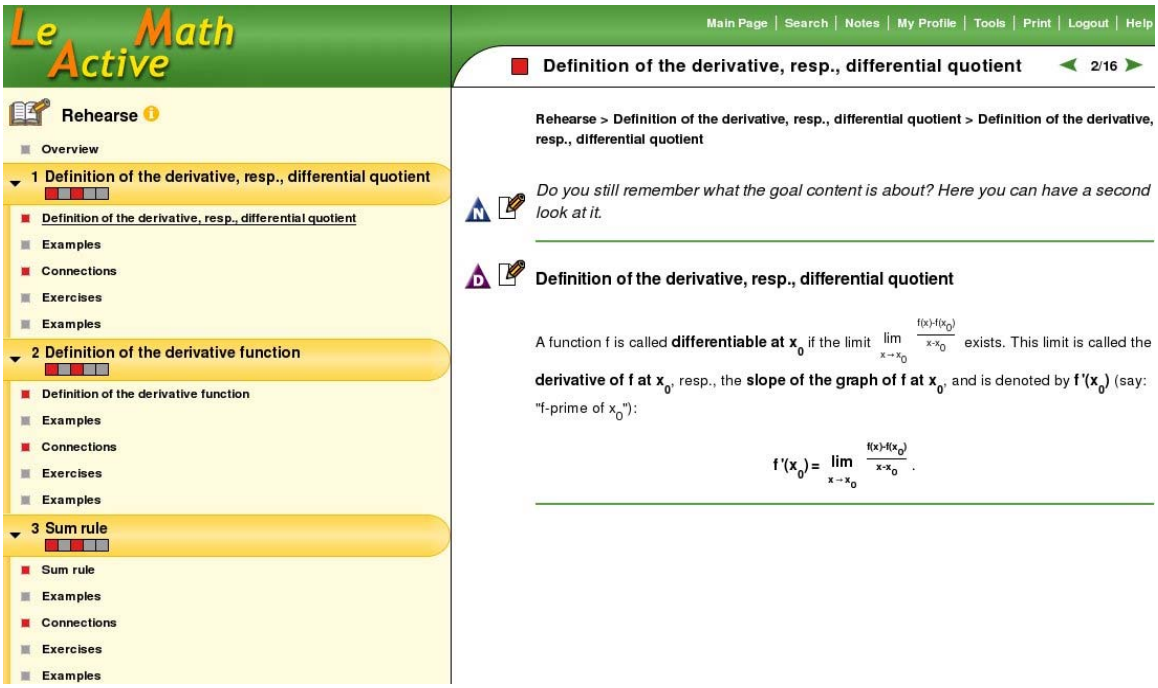

Figure 4. An example of a course of type *rehearse*

Figure 4 shows a course generated using the same concepts as the example shown in Figure 1, but for scenario *rehearse*. Note how the table of contents reflects the focus of the scenario on examples and exercises.

Scenario Connect

The scenario *connect* helps the learner to discover connections among the given concepts and other concepts and offers opportunities to train the concepts. The rationale of this scenario is that laws connect definitions by describing some relationship between the definition, for instance, laws in physics put physical concepts in relation to each other, and that becoming aware of these connections is beneficial to the user's learning (Novak & Gowin, 1984).

A course generated using the scenario *connect* is structured as follows:

1. Description. The course starts with a section that describes its aim and structure. Then, for each given concept, say, A, the following sections are inserted.
2. Present Concept. The concept is inserted, together with a concept map that shows its connections to laws and definitions.
3. Connect. For this section, all definitions (excluding A) are retrieved that are required by those theorems that require A. These are those definitions that are connected to A by theorems. Then, for each retrieved definition, say B, the following sections are created:
    a. Illustrate. This section presents example applications of the definition.
    b. Train. This section provides opportunities to train the definition.

    c. Develop. This section develops the definition in the following way:
        i. Present. The definition is presented.
        ii. Connect-2. This section presents all theorems that connect the original concept A with the definition B and all previously processed definitions.
4. Train-Connection. The course ends with a section that contains a concept map exercise.

Scenario Train Intensively

A course generated for the scenario *train intensively* generates a workbook that aims at increasing the competency level of the learning by presenting a large selection of exercises. The exercises cover all competencies and are presented with increasing difficulty level. The course structure consists of a flat sequence of exercise pages.

Scenario Train Competencies

A course generated for the scenario *train competency* trains a specific competency by presenting sequences of examples and exercises with increasing difficulty and competency level. The courses are structured as follows:

1. Description. This section provides a description of the scenario. Then, for each given concept, the following sections are created:
2. Rehearse. This section presents the current concept.
3. Illustrate and Practice. The following two sections are repeated for each competency level starting with the competency level one below the learner's current level:
    a. Illustrate. This section contains a sequence of examples of the competency level of the learner and of increasing difficulty level.
    b. Practice. This section contains a sequence of exercises of the competency level of the learner and of increasing difficulty level.

Scenario Exam Simulation

The scenario *exam simulation* contains exercises that can be solved within a specified timeframe. In contrast to the previous scenarios, the exercises are not selected with respect to learner properties since this is not the case in a real exam either. The generated courses consist of a number of pages, with each page consisting of exercises for the concepts selected by the user. The final page of the course contains an estimation of the amount of minutes it takes an average learner to solve all exercises.

Evaluations

Paigos was subject to Europe-wide formative and summative evaluation, which was performed by partners in the LeActiveMath project in Munich (Germany), Edinburgh (UK) and Malaga (Spain).

The summative evaluation focused on usability for the following reasons. First of all in complex e-Learning systems such as ActiveMath a number of factors occur that can not be separated properly. In our case scenarios and the results of the CG, always occur together with other factors such as interactive exercises and, of course, their benefit depends on the students' individual features. This makes assessing the benefits of CG alone very difficult. Even if the features etc. would not be as interacting, for a solid empirical evaluation a quite long empirical evaluation (several months) and large sets of users would be needed, split into a control group and an experimental group that gets the new conditions. This was impossible for us because we do not teach mathematics at a school or university ourselves. Only for very few intelligent tutoring systems such large-scale empirical studies were conducted, mostly for the cognitive tutors that are applied in many American schools (Koedinger et al., 1997). Recently, Brown et al., (2009) showed the difficulties of evaluating complex hypermedia systems: out of 24 systems examined, only two could be considered high quality studies.

Having said that, evaluations obviously still provide valuable data. For instance, the Technology Acceptance Model (TAM) (Venkatesh & Bala, 2008) formulates that perceived usefulness and perceived ease of use significantly determine intention to use and actual usage of technology. Positive results on these factors therefore allows us to draw conclusions about whether student will actually use a tool like Paigos in general and which scenarios in particular.

Two formative evaluations that took place in Germany helped us to discover and resolve some shortcomings of the formalized knowledge.

The summative evaluation was performed in January 2007 at the University of Edinburgh and University of Malaga. It involved 39 students of mathematics, engineering, science, and computer science, with an age range of 17–21 (average age of 18). 28 students were male, 11 female. On-line surveys were used to set the student a list of tasks (such as generating a course), to provide hints on how to solve the tasks, and ask for both structured (Yes/No or Likert scale questions) and for open feedback. At the time of the study, CG was new to the students and not part of their learning routine. The course was generated for content the subjects had worked with previously.

Several questions investigated Paigos' general usability. The subjects stated their agreement (100) or disagreement (0) using a 5-point Likert scale. The subjects highly value the CG wizard that guides the learners through the selection of the scenarios: they agree that it is easy to create a course (90) and that they are confident that they will find the content they want (77). Given that CG was previously unknown to the subjects, these results show that an adequate interface is able to convey the idea of CG to the learners.

An additional series of questions targeted the order and structure of the generated courses. About 70% students stated that the chapters were in the expected order. This relatively low value was caused by a programming bug, which under some circumstances resulted in chapters not being ordered with respect to the prerequisite relation. Even though this bug error was not planned from our side and is now corrected, its effects on the evaluation results are particularly interesting since they show that the students detect inappropriately ordered courses. Most subjects who disagreed with the ordering of the chapters reported this fact in the open feedback questions. Without the influence of the bug, the agreement ratio is 92%. The subject's agreement with the order of the pages is similarly high (92%). This indicates that automatic CG as implemented in Paigos structures courses in a way that students understand.

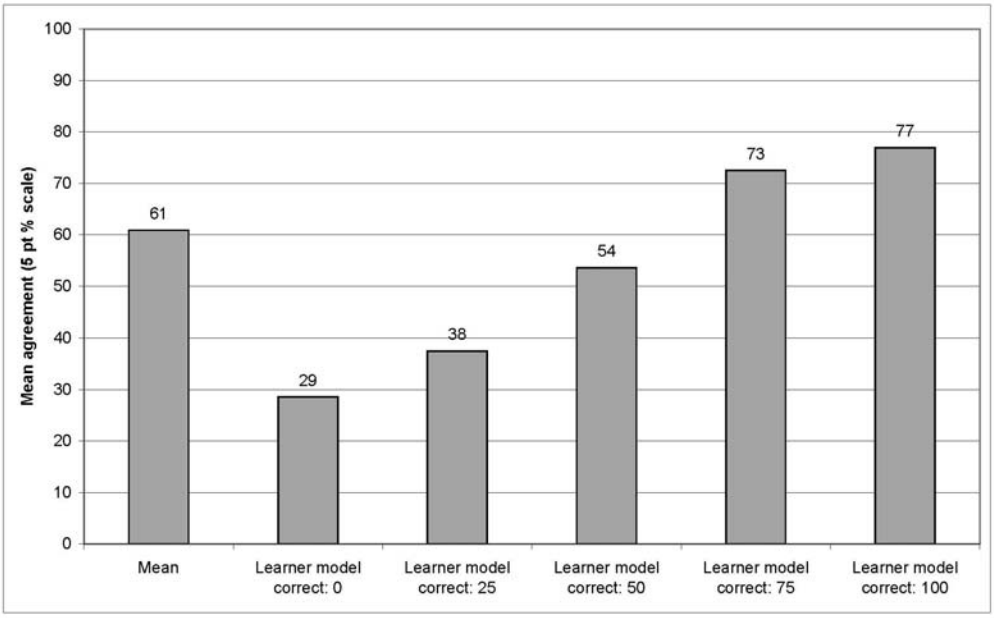

Figure 5. Results for the question "rate how well you think the content of the course has been tailored to your current knowledge", in relation to the perceived correctness of the student model

As shown in Figure 5, most subjects agree with the way Paigos adapts courses to their knowledge if they perceive their student model as correct. Students were asked to rate how well they think the content of the course has been tailored to their current knowledge using a 5-point Likert scale (0 not tailored at all, 100 perfectly tailored). The mean value of about 60 (first column) shows that the subjects agree with Paigos' adaptivity. However, a closer look on these figures shows that the rating of the quality of the personalization increases with the perceived correctness of the student model. One question in the evaluation asked the subjects to rate how well they think the student model reflects their competency. Columns 2 to 6 in Figure 5 show that the rating of the tailoring increases with the perceived quality of the student model. With a very low rating of the correctness of the student model, the quality of the tailoring is low, but not zero.

This may be due to the personalization with respect to the selected concepts. A high perceived student model quality (10 students for column 5, and 13 for column 6) increases the rating of the tailoring to almost 80. We draw two conclusions from this data: firstly, the learners are able to see that the courses generated by Paigos are adapted to the learner. Otherwise, the ratings would not differ depending on the rating of the student model. Secondly, the realized adaptivity is appreciated by the learners.

An additional question investigated the subjects' agreement/disagreement with the level of difficulty of the selected exercises. Among those students who stated that the estimations by the student model were correct, almost three quarters state that the selected exercises have the appropriate difficulty level (only a single subject disagrees). Again, this result indicates that Paigos' pedagogical knowledge is adequate from the students' point of view.

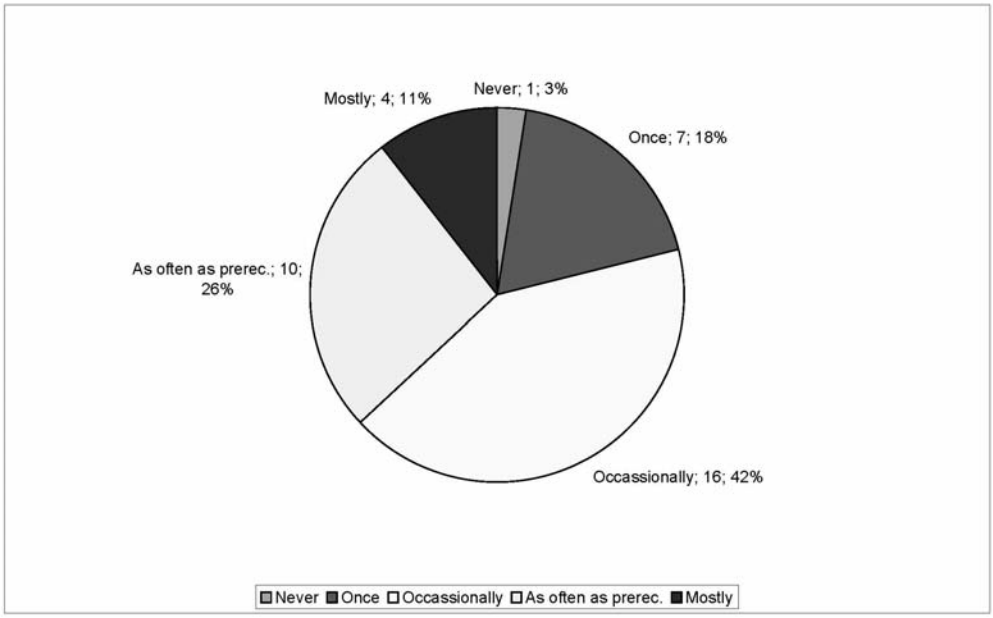

Figure 6. Results for the question "how much do you think you would use a discover course when learning a new topic?"

Subjects appreciate working with generated courses of the type discover. Figure 6 shows that only a single subject stated that he would never use a generated course. One third would use it mostly or as often as pre-authored course; forty percent would use it occasionally.

An additional set of questions investigated the subjects' attitudes towards the remaining scenarios. For each scenario, the subjects first read its description and then answered the question how much they think they could learn from generated course of this type. Subsequently, they generated the corresponding course. After examining the course, the subjects rated the scenarios by answering a series of questions.

The scenario *connect* was the least popular scenario. The elicitation of the connections between concepts is still uncommon in classroom teaching and hence unfamiliar to the learners. This is reflected in the neutral view (50) on whether one can learn from courses of this type after reading the scenario's description. Albeit the structure of connect courses is more complicated than of the other scenarios, students seem to understand the scenario (65). They have a neutral opinion (48) regarding the adaptivity. Interestingly, the subjects appreciate the scenario less after having seen a generated course than after having read the description (43 vs. 65). This indicates that the students value learning about connections between concepts, but that the actual realization needs to be refined: automatically generated concept maps that served to visualize the dependencies suffered from an extremely bad layout that confused more than it helped.

For the scenario *exam simulation*, the discrepancy between the agreement after reading the description (85) and after inspecting the course (54) is striking. We attribute this fact to the long time it took to generate this particular type of course. Due to technical reasons, several optimizations used in the other scenarios cannot be employed, which increases the generation time. The agreement regarding the comprehensibility of the structure is 82. 90% of the subjects would use courses of this type.

The subjects highly value personal courses of the type *rehearse*: the average agreement after reading the description and after generating a course is about 75. None of the subjects claimed he/she would never use a course of this type, about two third would use mostly or as often as pre-authored course. We attribute these results to the fact that rehearsing is a common task when learning and that students appreciate support during rehearsal, especially if the support is tailored to their competencies.

The appreciation of the scenario *train competency* is lowest of all scenarios (56) but increases after the subjects have inspected a generated course (61). We attribute these results to the fact that teachers and students are not familiar with the notion of "competencies". This would explain the increase: by only reading the description, the learners do not have a sufficiently concrete idea of the scenario. After inspecting a generated course, the subjects become aware that it mainly consists of exercises that train a concrete competency related to a concept. Two third of the students would use such a course at least occasionally.

The most appreciated type of course is *train intensively*. The mean agreement both before and after a course of this type was generated is about 80. About 85% of the subjects would use it at least occasionally, and half of the subject as least as often as a pre-authored course.

The evaluations show that the subjects understand the structure of the generated courses and appreciate the personalization. Those scenarios that support the students in working with exercises are valued highest. Scenarios such as connect that are new to students are rated lower.

The open feedback questions provided other insights. Two subjects indicated that they were afraid to miss important content when using generated courses: the generated course "was at a good level, but if I was revising for maths I would be worried that it would miss out something important, and just read the pre-authored [course] anyway". This underlines the importance of making the concept of generated courses familiar to the students if they are to be used in daily learning activities.

Another common response in the open feedback showed that a generated course is perceived as personal, that is, as tailored to and modifiable by the individual learner. In fact, all subjects agreed that they should be able to modify generated courses, a feature which is implemented in the ActiveMath. By using a menu attached to each educational resource, students can add content about this resource, e.g., additional examples, or remove the resource from the course. The subjects appreciate the options of modifying generated courses and the majority agrees that pre-authored courses should remain fixed ("kind of a backup"). Particularly interesting are the following two comments: the "freedom" of ActiveMath helps learners to find they own way of learning and makes them feel being respected, treated as adults.

> "One is responsible utterly for a generated course, but should always have a kind of backup in the form of pre-recorded course."
> "I think the more freedom a user is given, the more likely they will find the best way to learn the material for them."
> "As I understand, ActiveMath is designed for students; as students are adults, they should be able to easily decide about contents of their books, to suit their particular needs."

We conclude this section by discussing other relevant subjects' remarks (spelling errors were not corrected). The two following quotes illustrate that the subjects detected incorrect order of chapters, but were also aware of the intended structure in case of correct order.

> "differnce quotient is first but i think it should come later. pages are in a logical order."
> "They get progressively more difficult and the later ones require the knowledge of the first ones"

Three quarter of the subjects judged the difficulty level of the exercises as appropriate considering their competency level; several provided comments like the first quote below. The second comment might be caused by incorrect values in the student model.

> "the exercises are good as they grow in difficulty"
> "These problems seemed quite hard considering how weakly I have met the prerequisites."

The following four comments are typical for students' judgments on the benefit of pre-authored courses (i.e., non-dynamic courses that were authored by their teacher) and generated courses. Most subjects value the clear structure of generated courses and that they are focused on the selected target concepts. These quotes show that subjects perceive the adapted courses that Paigos generates as an added value compared to traditional access to content.

> "In generated courses topics are more friendly segregated."

"generated courses offers a better means of understanding the product rule as it gives information bit by bit ie in sequence"
"The pre-authored courses were set out ideally and in an acceptable order, I'd use that."
"I prefer generated courses because it is only about the product rule and no other topics would interrupt me. The pre-recorded course is quite big and is therefore not as clear as the generated course."

Conclusion and Future Work

This article presented a novel approach to CG that allows for the representation and application of complex pedagogical knowledge. Paigos shows that it is possible to formally encode pedagogical knowledge that is significantly more complex than in existing course generators.

Paigos is not tied to a specific learning environment. Due to Web-service interfaces and its usage of Semantic Web technology, its functionality is publicly available. For instance, Paigos is used by the learning environments ActiveMath (a learning environment used in several schools and universities European-wide as well as in Russia), MathCoach (Grabowski et al., 2005) and Teal (Rostanin et al., 2006). This openness helps to overcome the main limitation of CG, namely that it requires massive amounts of learning resources.

The pedagogical knowledge used for CG is specified as basic building blocks. From these, six different scenarios were formalized based on moderate constructivism. An additional scenario, not detailed in this article is based on Merrill's first principles of instruction (Merrill, 2002). This scenario served to demonstrate that the pedagogical knowledge formalized in Paigos can realize scenarios following different pedagogical paradigms. This illustrates the flexibility and pedagogical neutrality of Paigos.

The generated courses are structured according to pedagogical principles and contain automatically generated texts that describe the purpose of the sections. Together, these two features provide the student with information that may help them to understand why a course is structured in a specific way and to apply this knowledge when learning on their own.

Formative and summative evaluations of Paigos (actually the first in-depth evaluations of CG) show that students appreciate using a course generator, with varying degree of agreement depending on the scenarios, and perceive CG as an added value compared to traditional access to content.

The declarative representation of the CG knowledge and the modular architecture of Paigos ensure that extensions and new scenarios can be realized easily. This allows comparing different pedagogical scenarios (i.e., ways of presenting content and tools according to a specific learning goal) and pedagogical approaches (i.e., moderate constructivist courses vs. more traditional didactic approaches).

Thus, our hope is that Paigos can serve as a research platform that enables the formalization of pedagogical approaches, thus making them comparable. The evaluation of the different scenarios presented in this paper is only a first step in this regard. ActiveMath including Paigos is open source and available for download at http://www.activemath.org.

Acknowledgments

This work was funded by the EC (IST-2003-507826) and China PSF (No. 20080430656). The authors wish to thank Marianne Moormann, Tim Smith, Helen Pain, Ruth Hanson, Jeff Haywood and Hamish Macleod for performing the evaluations.

References

Brown, E. J., Brailsford, T. J., Fisher, T., & Moore, A. (2009). Evaluating learning style personalization in adaptive systems: Quantitative methods and approaches. *IEEE Transactions on Learning Technologies*, 2(1), 10–22.

Brusilovsky, P. & Vassileva, J. (2003). Course sequencing techniques for large-scale webbased education. *International Journal of Continuing Engineering Education and Lifelong Learning*, 13(1/2), 75–94.

Conlan, O ., Lewis, D., Higel, S., O'Sullivan, D., & Wade, V. (2003). Applying adaptive hypermedia techniques to semantic web service composition. In P. de Bra (Ed.), *Proceedings of AH2003: Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 53–62). Budapest, Hungary.

Grabowski, B. L., Gäng, S., Herter, J., & Köppen, T. (2005). MathCoach und LaplaceSkript: Ein programmierbarer interaktiver Mathematiktutor mit XML-basierter Skriptsprache. In K. P. Jantke, K.-P. Fähnrich, & W. S. Wittig (Eds.), *Leipziger Informatik-Tage*, volume 72 of *LNI* (pp. 211–218).: GI.

Karampiperis, P. & Sampson, D. (2005). Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology & Society*, 8(4), 128–147.

Kärger, P., Ullrich, C., & Melis, E. (2006). Integrating learning object repositories using a mediator architecture. In W. Nejdl & K. Tochtermann (Eds.), *Innovative Approaches for Learning and Knowledge Sharing, Proceedings of the First European Conference on Technology Enhanced Learning*, volume 4227 (pp. 185–197). Heraklion, Greece: Springer-Verlag.

Keenoy, K., Levene, M., & Peterson, D. (2003). *Personalisation and Trails in Self e-Learning Networks*. WP4 Deliverable 4.2, IST Self E-Learning Networks.

Koedinger, K., Anderson, J., Hadley, W., & Mark, M. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8, 30–43.

Melis, E., Goguadze, G., Homik, M., Libbrecht, P., Ullrich, C., & Winterstein, S. (2006). Semantic-aware components and services of ActiveMath. *British Journal of Educational Technology*, 37(3), 405–423.

Méndez, N. D. D., Ramírez, C. J., & Luna, J. A. G. (2004). IA planning for automatic generation of customized virtual courses. In *Frontiers In Artificial Intelligence And Applications, Proceedings of ECAI 2004*, volume 117 (pp. 138–147). Valencia (Spain): IOS Press.
First principles of instruction. *Educational Technology Research & Development*, 50(3), 43–59.
Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research*, 20, 379–404.
Novak, J. & Gowin, D. (1984). *Learning How to Learn*. New York: Cambridge University Press.
Pintrich, P. R. (1999). The role of motivation in promoting and sustaining self-regulated learning. *International Journal of Educational Research*, 31, 459–470.
Prenzel, M., Drechsel, B., Carstensen, C. H., & Ramm, G. (2004). PISA 2003 - Eine Einführung. In PISA-Konsortium Deutschland (Ed.), *PISA 2003 - Der Bildungsstand der Jugendlichen in Deutschland - Ergebnisse des zweiten internationalen Vergleichs* (pp. 13–46). Münster, Germany: Waxmann Verlag.
Reinmann-Rothmeier, G. & Mandl, H. (2001). Unterrichten und Lernumgebungen gestalten. In A. Krapp & W. Weidmann (Eds.), *Pädagogische Psychologie. Ein Lehrbuch* (pp. 601–646). Weinheim: Beltz PVU, 4.edition.
Rostanin, O., Ullrich, C., Holz, H., & Song, S. (2006). Project TEAL: Add adaptive e-learning to your workflows. In K. Tochtermann & H. Maurer (Eds.), *Proceedings: I-KNOW'06, 6th International Conference on Knowledge Management* (pp. 395–402). Graz, Austria.
Sicilia, M.-A., Sánchez-Alonso, S., & García-Barriocanal, E. (2006). On supporting the process of learning design through planners. In *Virtual Campus 2006 Post-proceedings. Selected and Extended Papers* (pp. 81–89).
Specht, M., Kravcik, M., Pesin, L., & Klemke, R. (2001). Authoring adaptive educational hypermedia in WINDS. In N. Henze (Ed.), *Proc. of the ABIS 2001 Workshop*.
Specht, M. & Oppermann, R. (1998). ACE - adaptive courseware environment. *The New Review of Hypermedia and Multimedia*, 4, 141–162.
Tergan, S. O. (2002). Hypertext und Hypermedia: Konzeption, Lernmöglichkeiten, Lernprobleme und Perspektiven. In P. Klimsa & L. Issing (Eds.), *Information und Lernen mit Multimedia und Internet – Lehrbuch für Studium und Praxis* (pp. 99–112). Weinheim: Beltz Verlag.
Ullrich, C. (2005). The learning-resource-type is dead, long live the learning- resource-type! *Learning Objects and Learning Designs*, 1(1), 7–15.
Ullrich, C. & Melis, E. (2009). Pedagogically founded courseware generation based on HTN-planning. *Expert Systems With Applications*, 36(5), 9319–9332.
Van Marcke, K. (1992). Instructional expertise. In C. Frasson, G. Gauthier, & G. McCalla (Eds.), *Proceedings of the Second International Conference on Intelligent Tutoring Systems, Montréal, Canada*, volume 608 of *Lecture Notes in Computer Science* (pp. 234–243). Heidelberg: Springer.
Vassileva, J. & Deters, R. (1998). Dynamic courseware generation on the WWW. *British Journal of Educational Technology*, 29(1), 5–14.

Venkatesh, V. & Bala, H. (2008). Technology acceptance model 3 and a research agenda on interventions. *Decision Sciences*, 39(2), 273–315.
Wiley, D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D. A. Wiley (Ed.), *The Instructional Use of Learning Objects: Online Version.*
Zech, F. (2002). *Grundkurs Mathematikdidaktik*. Weinheim: Beltz Verlag.