Investigating the Reuse of Course Generation Knowledge

Carsten Ullrich[a/b], Qinghua Chen[b], Sushing Chen[c], Liping Shen[b], Ruimin Shen[b]
[a]*DFKI GmbH, Germany*
[b]*Computer Science Department, Shanghai Jiao Tong University, China*
[c]*PICB, Shanghai, China*
cullrich@activemath.org

Abstract: Course generation knowledge, i.e., how to assemble a sequence of learning objects which is adapted to a learner's goals, competencies and preferences, is hard to assess and expensive to formalize, at least if it is based on pedagogical principles. On the other hand, once a course generator has been developed, it allows generating personalized courses for individual users at a relatively low cost. Therefore, course generation lends itself of being available as a (Web) service such that other systems can access its functionality without having to implement the knowledge themselves. In this paper we investigate whether this reuse of course generation knowledge works in practice. We took a course generator that was developed in the European project LeActiveMath and made it available for use in a Chinese distant university. In the paper, we describe how we represent the generic course generation knowledge and the difficulties we faced and the results we achieved when we applied the course generator in the Chinese setting. We also report on using course generation in a bioinformatic workflow environment.

Keywords: course generation, metadata, reuse, pedagogical knowledge

Introduction

In this paper, we investigate the reusability of pedagogical knowledge employed for course generation. Shortly speaking, a course generator (CG) takes a user (or learner) identifier and a learning goal as input and generates a structured sequence of learning objects (LO) that is adapted to the learners' competencies, individual variables and learning goals. CG is seen as one of the most interesting research areas in courseware management and also highly relevant from a commercial point of view "since [course generation] will enable a course provider to accommodate the needs of different customers at a fairly small cost" [1].

The pedagogical knowledge that is formalized within the CG determines which learning goals it can process: the more sophisticated the CG knowledge, the greater the range of learning goals and realized adaptivity. However, formalizing pedagogical knowledge is resource-intensive work: as an example, *Paigos* was developed and refined over a period of three years. It thus is reasonable to make the CG knowledge available as a service to other systems. This way, third-party learning environments that want to offer CG to their users do not require their own implementation of the required pedagogical knowledge but can access a CG service and make use of it. However, the realization of a CG service faces several difficulties. *Generic CG Knowledge*: the pedagogical knowledge formalized with the CG needs to be independent of the domain that is taught. Otherwise, it might not be applicable to the domain of the client. *Repository Integration*: how can the client connect its own LO-repository to the CG service? *Heterogeneous Metadata*: typically each LO-repository uses its own metadata to describe the LO, despite standardization efforts. However, the CG needs to reason about the LO that are stored within the client's repository (e.g., it needs to know whether there exists an easy example for a given concept).

In this paper, we describe the techniques we employed to solve these difficulties in

Paigos and illustrate the practical applicability using an example application that takes place in an environment significantly different from the one *Paigos* was developed for.

We start by a short overview on CG and on related work. Then, Section 2 describes *Paigos* with a focus on the techniques relevant for CG as a generic service. Previous usages of the CG service were closely related to the setting *Paigos* was originally developed for: Web-based learning support for European high-school and first year university students. The environment described in this paper (Section 3) is significantly different, namely the Network Education College (NEC) of Shanghai Jiao Tong University. There, differences are technical as well as cultural: lectures at NEC consist of a mixture of video lessons, PowerPoint slides and HTML resources in contrast to the "pure" Web-based environments *Paigos* was previously applied in; culturally because the learning needs of full-time students and vocational learners differ as well as the teaching methods. The last part of Section 3 describes the usage of CG in an even more different setting, namely a workflow environment for bioinformatics. The article ends with a short conclusion.

1. Related Work

Course generation has long been a research topic [2]. It uses pedagogical knowledge to generate a structured sequence of learning objects that is adapted to the learners' competencies, individual variables, and learning goals [1]. This generation happens upon request of a client (a learner or a software system). Ideally, the sequence is not a flat list of learning objects but is structured in sections and subsections. This structure can convey additional information relevant to the learning process. In course generation, the course is generated completely before it is presented to the learner. This early generation has the advantage that the course can be visualized to the learner, thereby informing her about the structure. In addition, the student can navigate freely through the course. Course generators can be described using the Adaptive Hypermedia Application Model [3]. A CG consists of a domain model, a user model and a pedagogical model. The domain model contains the LOs and, depending on the specific system, the domain concepts. Additional information associated to the domain model includes metadata and the domain structure. Based on observations of the user's interactions, the user model (or learner model) stores, infers and updates information about an individual user. The pedagogical model contains the knowledge how to adapt the behavior of the system, e.g., how to present content from the domain model taking into consideration the information provided by the user model.

Early work on CG [4] emphasized the value of the teaching knowledge used by the pedagogical module. Today's CG [5] [6]often uses rather simplified pedagogical knowledge that guides the assembly, most of the time the "prerequisite"-relation that represents that understanding resource B depends on having understood resource A is used. One reason for using only very simple pedagogical knowledge is that pedagogical knowledge is hard to assess and expensive to model. Systems that aim at modeling pedagogically knowledge require a large teaching model: for instance, the instructional knowledge base of the system GTE encompasses about a hundred tasks and methods; the instructional ontology proposed by [7] consists of about 530 classes. *Paigos*, the CG described in this article in Section 2, contains about 300 rules that determine the selection, ordering and structuring of LOs. This "expensive" functionality lends itself to being "outsourced": if the CG is available as a service then other learning environments can access the functionality without having to re-implement it. However, none of the previously developed CG allowed the integration of new repositories and the accessible content was restricted to the local repositories. *Paigos* in contrast is a Web service where clients can register their repositories and subsequently use *Paigos* for CG. The next section discusses this in more detail.

2. The Course Generator *Paigos*

Paigos is a CG that was developed in the European FP6 project LeActiveMath. *Paigos* is used to generate courses that support a learner in achieving a number of learning goals, such as discovering new content ("discover" in short), training specific competencies and simulating exams. For these learning goals, *Paigos* generates complete courses which contain all the learning material required by a learner to achieve the goals. *Paigos* is also used to retrieve single elements that specifically fulfill a given purpose, such as presenting an example or exercise adequate for the current the learner. This functionality is important for remedial, e.g., if a learner fails to solve an exercise, then the presentation of an example might help the learner to overcome the difficulty.

Paigos was developed in a moderate constructivist context. Moderate constructivism emphasizes the active engagement of the learner and stresses that knowledge cannot be transferred from the teacher to the learner but is the result of cognitive processes in the learner's mind. A moderate constructivist learning environment has to stimulate and create opportunities for these processes. *Paigos* is also based on the concept of "competencies", which advocates that different competencies together build up mathematical literacy [8]. In the following, we describe the AI (Artificial Intelligence) framework employed by *Paigos* and how it is used to describe two scenarios ("discover" and "guided tour").

2.1. Hierarchical Task Network Planning

Paigos uses AI planning as a framework for implementing the pedagogical knowledge of how to generate a course. In HTN planning [9], 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 these top tasks into smaller subtasks until *primitive* tasks are reached that can be carried out directly using *operators*. The *planning operators* describe various actions that can be performed directly. *Methods* describe various possible ways of decomposing non-primitive tasks into subtasks. These are the "standard operating procedures" that one would normally use to perform tasks in the domain. Each method may have constraints that must be satisfied in order to be applicable. Planning is done by applying methods to non-primitive tasks to decompose them into subtasks, and applying operators to primitive tasks to produce actions. If this is done in such a way that all of the constraints are satisfied, then the planner has found a solution plan; otherwise the planner will need to backtrack and try other methods and actions.

For CG, *Paigos* takes a *goal task* as input and returns a structured sequence of LO identifiers (basically a table of contents) as a result [10]. The goal task consists of an educational objective that represents the type of learning goal and of learning objects identifiers for which this learning goal should be achieved. As an example, the goal task "discover 'average slope'" represents the goal of a learner who want to reach an in-depth understanding of the mathematical concept 'average slope'. We now take a closer look on the knowledge that formalizes how to achieve such a learning goal.

2.2. Scenarios "Discover" and "Guided Tour"

The scenario "discover" generates courses that contain those LOs that support the learner in reaching an in-depth understanding of the concepts given in the goal task. The basic structure of the scenario follows a course of action in a classroom as described by [11] that consists of several stages that typically occur when learning a new concept. For each stage, the course contains a corresponding section. The following sections are created:

- Description: describes the aim and structure of a course. Then, for each concept given

in the goal task, the following additional sections are created:

- Motivation: motivates the usefulness of the concept using adequate LOs (the precise meaning of an "adequate" LOs is formalized in methods like described below). It also contains the unknown prerequisites.
- Develop: presents the concept and illustrates how it can be applied.
- Train: provides opportunities to train the concept.
- Connections: illustrates the connections between the current and related concepts.
- Reflection: each course closes with a reflection section, which provides the learner with opportunity to reflect on the content presented in the course.

For this scenario, the course generation is started with the single goal task `(discover (c_1...c_2))` with `c_1...c_2` representing the goal concepts the student wants to learn. A first method, not shown here, inserts a subtask `(learnConceptDiscover (c_n))` for each of the concepts. This task is decomposed by the following method:

```
(:method (learnConceptDiscover ?c)
  ()
  ((!startSection)
   (introduceWithPrereqSection ?c)
   (developConcept ?c)
   (proveSection ?c)
   (train ?c)
   (showConnectionsSection ?c)
   (!endSection)))
```

The keyword `:method` starts the definition of a new method. The subsequent expression `(learnConceptDiscover ?c)` represent the task the method is applied on (expressions starting with "?" stand for variables and are instantiated at run-time). The task is decomposed into 7 subtasks `(!startSection)`... `(!endSection)`: first, a new section is started, in this case the course itself. Then, several subtasks are inserted that mimic the informal description above. The last subtask closes the course. Similar methods exist for the other tasks.

About 30 methods encode the pedagogical knowledge how to select exercises that are "appropriate" for the learner. The exact meaning of "appropriate" differs depending on the individual learner. The most relevant factors determining exercise selection are the educational level of the learners and their current competency level. In general, the *learning context* of each LO should correspond to the educational level of the learner. Otherwise it may be inadequate, that is, either too simple or too difficult (think of a second year university student being presented a definition for elementary school). In addition, in most cases, resources presented to the learner should be of a competency level that corresponds to the learner's since these are the resources he is able to master. In some situations resources with a higher competency level need to be selected, e.g., when aiming at increasing the competency level. The methods also take information about motivation and anxiety into account (assed using the performance in exercises). A similar complex set of methods determines the selection of examples.

The scenario "discover" is based on competencies and competency levels. Since competency-based pedagogy is a relatively novel approach, it is not yet widespread and most of the existing learning environments follow the traditional mastery-based paradigm. With this in mind, we developed the scenario "guided tour" based on Merrill's "First principles of instruction" [12]. In this scenario, for each concept given in the goal task, and for each unknown prerequisite concepts, the following sections are created: An introduction that arises a learner's interest by presenting LOs of the type introduction; a section that presents the concept itself, a section that provides explaining and deepening information about the concept; a section that provides opportunities for the learner to examine demonstrations of applications of the concepts; a section that enables a student to actively apply what he has learned; and finally a section that contains concluding information about the concept.

2.3.Course Generation Web-Service

Paigos makes its functionality available as a Web-Service (CGWS). Clients send a learning goal to the CGWS and receive a structured sequence of LO identifiers as a result. The sequence is represented using a standard representation called IMS Manifest [13]. The CGWS implements a mediator architecture that enables a client to easily register its own repository and thus make it available to *Paigos*. At registration time, a client has to provide the name and the location of its repository, together with an ontology that describes the metadata structure used in the repository and a mapping of the metadata used in the CG onto the repository ontology. The metadata in *Paigos* consists of an ontology of instructional objects that describes LO from a pedagogical point of view sufficiently precise in order to allow for intelligent automatic pedagogical services [14].

3. Applying Course Generation in a New Context

The Network Education College (NEC) is Shanghai Jiao Tong University's distant university. Founded in 2001, today NEC has more than 15,000 students. The students are vocational learners: they work during the day and study in the evening and on the weekend. The large number of users makes it impossible to provide individualized human-human support. Therefore, research at NEC focuses on intelligent *automatic* support. In theory, such a setting is ideal for the personalization offered by CG. However, in practice we had to overcome several difficulties. The first one consisted of determining how to integrate best the CG in the pedagogical approach employed by NEC. In ActiveMath the pedagogical paradigm was to give as much freedom to the learners as possible and allow them to define their learning goals on their own. The students were full-time university and school students, and thus could concentrate on learning and take their time in exploring the content. In NEC students are typical vocational learners: they have limited time available for study, need to focus and be able to quickly work through the curriculum. Thus, we had to identify potential points of interest for CG usage. This “organizational integration” is described first, followed by a discussion of the technical aspects of the integration.

3.1.Organizational Integration

Current research at NEC investigates among others the identification of the concepts a student needs to learn in order to achieve a learning goal and how to support students while working through a course. We describe both of them in detail.

At NEC, a formal representation of the subject domain and its interrelationships is given by a domain ontology. Attached to the concepts represented in the ontology are the *competencies* that a learner needs to master in order to understand the concepts (the competencies are different from those used in ActiveMath). LOs are defined as being instances of the domain concepts and are annotated using a subset of LOM [15]. Based on these representations, the competencies that a learner needs to acquire are identified using sets of rules similar to the sequencing rules of the IMS SS standard [16]. This is achieved by a competency gap analysis that determines which competencies the learner needs to learn to fill the knowledge gap (the algorithm is presented in [17]). At this time, CG can come into play: the inputs to CG are the concepts that are linked to the missing competencies. The output is a course that contains the LO that support the learner in understanding the goal competencies. This way, with the push of a button learners have access to a targeted course adapted to their individual knowledge state, without any additional load that may distract their learning.

The second opportunity to employ CG is for teacher support. A *teacher client* was developed to help the teacher when interacting with large amount of learners who have dif-

ferent knowledge states. It divides the students into different clusters according to their learning ability, thereby significantly reducing the workload of the teachers since they no longer have to address individuals but groups. In case a group exhibits potential learning problems, the teacher recommends LOs to the students of the group. CG can support this process by selecting the specific resources that are needed by the groups. In this way, teachers only decide about the general learning goal, say, "present example for concept x" and *Paigos* will select the appropriate example for this group of learners automatically.

3.2. Technical Integration

The technical integration consists of two steps: first, the client registers its repository at the CGWS in order to make its LOs accessible during course generation. Second, the client accesses the CG itself, by passing a learning goal and receiving a structured sequence of LOs as a result. *Paigos* makes both functionalities available by a Web-service interface. The goal of the client repository registration is that *Paigos* can send queries about LOs to the client repository. During CG *Paigos* needs to know, say, whether there exist difficult exercises for a concept X, what are the prerequisites of concept X... These queries are used by the methods and operators of *Paigos* to decide which LOs to include in a course. They are formulated using a vocabulary formalized in an ontology of instructional objects [14]. The ontology consists of 26 different classes each representing a specific pedagogical type a LO can have (e.g., definition, example, process, fact, exercise, proof ...). These classes together with several relations define a set of *metadata* that is required for automatic pedagogical decision-making as performed by *Paigos*. Since in practice each repository *R* uses its own representation of metadata, the queries sent by *Paigos* need to be translated into the terms used by *R*. This is realized by a mediator architecture and ontology mappings: the mediator requires ontological descriptions of both metadata structures and a mapping between them. Then, it translates queries from *Paigos* into the format required by *R*.

The situation at NEC is representative for today's commercial learning environments: since annotating content with metadata is an expensive task, metadata is rarely used. NEC distinguishes only between three types of LOs. This makes it of course difficult for *Paigos* to select resources intelligently: if it cannot distinguish between different LOs then it cannot insert LOs of a specific type. Nevertheless we decided to apply CG at NEC since it would serve as a representative use case to see whether the pedagogical knowledge degrades gracefully, that is whether it generates sensible courses with limited metadata. We decided against adapting the rules to the local context since a Web-service should enable a client to use its functionality without requiring changes in the server for each potential client.

At NEC, three types of LOs are stored in an SQL-database: *soc_resources* (SOC_R), *soc_knowledge_points* (SOC_K) and *soc_exercises* (SOC_E). SOC_Ks correspond to concepts. They have a name and a relation that encodes the dependencies between SOC_Ks. SOC_Es correspond to exercises. A relation encodes which SOC_K an exercise trains. SOC_Rs are descriptions of SOC_Ks. They can be HTML pages, Powerpoint slides and video-lectures. A relation encodes which SOC_Ks are explained by a SOC_R. Both SOC_E and SOC_R have a difficulty rating, and each SOC_R has an importance value. Despite this limited amount of different type, there exist several potential mappings from the CG ontology on the NEC ontology. In a first approach, we decided for potentially over-simplistic solution: SOC_K is mapped onto *concept* (called "fundamental" in the CG ontology) and SOC_E is mapped onto *exercise*. The mapping of SOC_R was harder to decide. SOC_Rs contain definitions, application examples as well as general information. Therefore, in a first step we decided to map it onto the general class *remark*.

The ontology and the mapping being defined, we had to implement the Web service interfaces required at the client side. The mediator requires that a repository answers queries

ask for a) the type of the given learning object; b) all the LOs that are connected to a given LO by a given relation; c) all properties the given LO has. The method `registerRepository` registers the repository that the client wants the course generator to use. The client has to provide the name and the location of the repository. Additional parameters include the ontology that describes the metadata structure used in the repository and the mapping of the CG ontology onto the repository ontology.

Finally, using the method `generateCourse`, the NEC client starts the course generation on a learning goal it passes to *Paigos*. Since CG adapts the courses to individual learners, *Paigos* needs access to information about the learner. In general, information about the individual learner can be made available to *Paigos* in two ways: if the client contains a learner model that stores information about learners, then the client can register a learner model and pass the respective learner identifier as a parameter. In case no learner model exists, a client gives a list of property-value pairs that is used by the *Paigos* to construct a temporary "learner model": the course generator performs the planning in the same way as with a real learner model; however its access of learner properties is answered using the map. Properties not contained in the map are answered with a default value. In the current version of *Paigos*, the remote access of the learner model is not yet implemented. Therefore, the NEC system passes information about the individual learner (the first integration case) or about the group of learner (the second integration case) to *Paigos*.

3.3. Results

We applied the CG to learning materials about *data structures*. Typically, the associated lecture is attended by about 230 students. We started Paigos on several learning goals and asked the lecturer to comment the generated courses. Since the scenario "discover" significantly relies on metadata which is not available at NEC, we had to define the mapping in a way such that this metadata was ignored. As a result, the courses were rated being of medium quality. However, the lecturer significantly appreciated the time savings made possible by the automatic generation.

The fact that at NEC courses consist of a mixture of HTML, slides and video-lecture posed no problem to Paigos. The pedagogical knowledge is designed to abstract from the used media type and focuses on the instructional purpose of the LOs. The resulting courses are presented as a table of contents to the learner and following the contained references (e.g. by clicking on the link) opens the associated viewer.

3.4.Learning in a Bioinformatics Workflow Environment

A workflow workbench environment [18] allows a user to specify and execute scientific workflows. The workflows model "in silicio" experiments: they involve the combination of data and analyses made available by the research teams. In bioinformatics, this type of experiments complements lab-based experiments and allows to generate new information from the publicly available data and to form hypothesis which can be assessed in lab studies. Workflows enable a scientist to model and execute their experiments in a repeatable and verifiable way. Just like experiments can be repeated by other research groups, they can be exchanged and are resources in their own right. Thus, workflows capture the essential aspects of "in silicio" experiments and, what makes them so valuable for learning, they make tacit procedural knowledge explicit. They thus offer a way of communicating this knowledge to students. A workflow essentially consists of input and output nodes and of processor nodes that perform operations on data. For learning, a processor node can be seen as an example of the application of the service it represents. Thus, when studying a workflow, a

learner should be able to select a processor node and request educational resources that elaborate on this service and its specific operation. The request is processed by the CG. This requires a corresponding set of resources that are annotated with the service they illustrate. Resources can thus play a double role. They describe a specific phenomenon and at the same time they are examples of the usage of the service they were created with.

4.Summary and Conclusion

Course Generation knowledge can be designed to be generic and reusable, even in a significantly different context than it was developed in. The quality of the generated courses obviously depends on the metadata that is available in the repositories used by CG: if only a limited amount of information about the LOs is available, then the system has no means to distinguish between the resources and to select the most appropriate one. Still, even with limited information, the generated courses were evaluated as being usable. The advantage of a CG service is that as soon as additional information about the LOs is available (i.e., the metadata is extended) and the CG is informed of it (by a changed ontology mapping), it is used by CG. Once the LOs are extended with additional metadata, the client only needs to adapt its ontology and mapping and re-register its repository at the CG. Then, newly registered courses will take this additional knowledge into account. The client does not need to take care of the pedagogical knowledge; the CG takes care of it.

References

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

[2] D. R. Peachy and G. I. McCalla. Using planning techniques in intelligent tutoring systems. *International Journal of Man-Machine Studies*, 24(1):77–98, 1986.

[3] P. De Bra, G.-J. Houben, and H. Wu. AHAM: a Dexter-based reference model for adaptive hypermedia. In *Proceedings of HYPERTEXT'99*, pages 147-156, New York, NY, USA, 1999. ACM Press

[4] K. Van Marcke. GTE: An epistemological approach to instructional modelling. *Instructional Science*, 26:147–191, 1998.

[5] K. Keenoy, M. Levene, and D. Peterson. Personalisation and trails in self e-learning networks. WP4 Deliverable 4.2, IST Self E-Learning Networks, 2003.

[6] N. D. D. Méndez, C. J. Ramírez, and J. A. G. Luna. IA planning for automatic generation of customized virtual courses. In *Proceedings of ECAI 2004*, pages 138–147, Valencia (Spain), 2004. IOS Press.

[7]Y. Hayashi, J. Bourdeau, and R. Mizoguchi. Ontological support for a theory-eclectic approach to instructional and learning design. In *Proceedings of EC-TEL 2006*, pages 155–169. Springer-Verlag, 2006

[8] OECD, editor. *Learning for Tomorrows World — First Results from PISA 2003*. Organisation for Economic Co-operation and Development (OECD) Publishing, 2004.

[9] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. Shop2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[10] C. Ullrich. Course generation based on HTN planning. In *Proceedings of 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems*, pages 74–79, 2005

[11] F. Zech. *Grundkurs Mathematikdidaktik*. Beltz Verlag, Weinheim, 2002.

[12] M. D. Merrill. First principles of instruction. *Educational Technology Research & Development*, 50(3):43–59, 2002.

[13] IMS Global Learning Consortium. *IMS content packaging information model*, 2003.

[14] C. Ullrich. The learning-resource-type is dead, long live the learning- resource-type! *Learning Objects and Learning Designs*, 1(1):7–15, 2005.

[15] IEEE LTSC. 1484.12.1-2002 *IEEE standard for Learning Object Metadata*, 2002.

[16] IMS Global Learning Consortium. *IMS simple sequencing specification*, 2003.

[17] L. Shen and R. Shen. Ontology-based Intelligent Learning Content Recommendation Service. *International Journal of Continuing Engineering Education & Life-Long Learning*, 15(3-6):308-317, 2006.

[18]Oinn, T., Greenwood, M., Addis, M., Alpdemir, M. N., Ferris J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M. R., Senger, M., Stevens, R., Wipat, A. and Wroe, C.. Taverna: Lessons in creating a workflow environment for the life sciences in Concurrency and Computation: Practice and Experience, Volume 18 Issues 10, pages 1067-1100, *Grid Workflow Special Issue*, August 2005