A Distributed Ontological Approach as a Basis for Software in the Context of Academic Programs

Richard Hackelbusch and H.-Jürgen Appelrath

Carl von Ossietzky Universität Oldenburg,
Escherweg 2, 26121 Oldenburg, Germany
hackelbusch@uni-oldenburg.de
http://www-is.informatik.uni-oldenburg.de/

Abstract. The implementation of a computer understandable representation of the semantics of academic programs is complex. That's why academic institutions struggle in implementing pervasive information systems that offer services to help all actors in this context. These services are demanded by, *e. g.*, students who want to plan their curricula correctly, or who want to know which courses can be used for a different academic program or at a different academic institution. In this paper, we introduce a distributed ontological approach to represent the semantics of academic programs and their examination regulations, the universities' supply, and individual results. It allows academic institutions to implement applications that offer the demanded services and that use these ontologies as a common basis.

1 Introduction

Written in a legal language, academic institutions release examination regulations and subsidiary documents that describe their academic programs. Because legal language is very hard to comprehend by humans and in this form not interpretable by computers, a great demand for decision support exists. In this context, unfortunately, there is only little automatic help — like decision support systems — realized. Results of this situation, *e. g.*, are that students often do not understand the whole descriptions of academic programs or even do not try to read them. Thus, they have questions concerning good and correct ways in planning and realizing their individual curricula. Examples are which offered courses they can take at best or which of them can be used for a different academic program, *e. g.*, in the case of a minor subject. Another interesting question is, *e. g.*, which courses can be taken for academic programs of another academic institution. Besides the computer-understandable representation of the semantics of examination regulations, another challenge is the distributed structure of academic institutions and of clusters of them. Information concerning academic programs and their examination regulations, the universities' supply of courses and the individual results of the students is often created and stored separately

P. Dillenbourg and M. Specht (Eds.): EC-TEL 2008, LNCS 5192, pp. 122–127, 2008.

by, *e. g.*, faculties and institutions (see [1]). It has to be merged in order to be able to offer decision support applications.

In this paper, we introduce an ontological approach for the representation of academic programs and their examination regulations, the universities' supply of courses and the individual results of the students. This approach allows a distributed representation that fits to the structure of academic institutions. We explain that our approach is a very applicable basis to implement applications offering decision support in the context of academic programs.

2 The Ontological Approach

In order to represent the semantics of academic programs and their examination regulations, we have identified two important aspects: On the one hand, there is a *static* representation needed which defines the concepts of *entities* of academic programs — a conceptual model. An exemplary representation would be, *e. g.*, *modules* that can contain a couple of *courses* and *examinations*; *courses* can be used for certain *modules*, deal with specific *topics* and so on. These entities can be structured very heterogeneously comparing different academic programs (see [2]). On the other hand, there is a *dynamic* representation needed which defines a possible order of concrete entities in the course of a study. In addition, it defines conditions that regulate the possibilities in taking certain of these concrete entities. It also defines, how an entity (like a *module*) can be successfully passed, *e. g.*, by taking a couple of *courses* and calculating the average grade which has to be better than a specific grade.

The *static representation* can be modeled well using an object-oriented or an entity-relation based representation. It is a conceptualization that describes the static structure of all possible instances. The availability of a common shared conceptualization would be very useful because we have to deal with a distributed structure of academic institutions (and of clusters of them). Then, different faculties or institutions can instantiate these concepts, *e. g.*, modeling their own supply and share this information with other faculties or institutions on the web. In addition, the examination offices could use the conceptualization and these instances for representing the individual results of their students. Beside others, these aspects motivate our decision to use ontologies to represent the static representation of academic programs. The *dynamic representation* covers rules in taking and passing instantiations of the concepts that have been defined in the course of the conceptual modeling of the static representation. To do so, academic programs and their examination regulations can be represented as a kind of process (see [3]).

Our ontological approach contains a general meta-model (implemented in OWL-DL) with a set of business-rules and a software framework (see [4]). We call the approach Curricula Mapping Ontology (CMO). The framework includes a library that uses the JENA-framework (http://jena.sourceforge.net/). It can interpret instantiations of the meta-model and it ensures that they stick to the business-rules. The meta-model is a conceptual model that defines concepts that

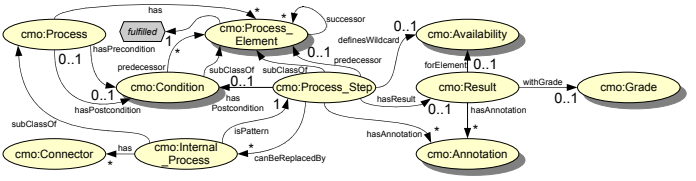

Fig. 1. Basic process and wildcard concepts of the CMO

allow the instantiation of processes including conditions that represent very complex rules — if needed. These processes represent academic programs and their examination regulations. It is intended to extend specific concepts of the general meta-model. This extended conceptual model is the static representation of academic programs. Instantiations of this individually extended meta-model form the dynamic representation of academic programs including their examination regulations. They also form representations of the universities' supply, and the individual results of the students. Given a set of individual results of a single student, the model-interpreting framework can, *e. g.*, show the possibilities of the student in continuing his academic program. Our approach is conceptualized that way that nearly arbitrary extensions of the general meta-model are allowed to define the static representation of an academic program without the need of adapting the model-interpreting framework. As long as the business-rules of the general meta-model are fulfilled, the framework can interpret all possible instantiations (the dynamic representations) of such extended meta-models synchronizing it with a set of individual results of a student.

The general meta-model of the CMO defines concepts to represent processes (see figure 1 — we use the namespace shortcut "cmo"). A process contains a set of process elements (the class Process_Element is not intended to be instantiated directly; it has to be specialized — this is shown by shadows). These elements can be conditions or process steps. Each element can have predecessors and successors in order to be able to create an order/a process. Conditions can be interpreted as TRUE or as FALSE depending on the type of their specialization and the values of their preceding elements. They can be, *e. g.*, simple logical conditions like AND — known from logic gates. They also can be more complex conditions that include the comparison of aggregations of certain values of attributes of their preceding elements with other values. An example is: "the number of passed elements > 6". It is also possible to aggregate values of the preceding elements that are self-defined within the static representation and use it in connection with a comparison like "the sum of *workload* > 16" (see [4]).

Each process step references a specialization of Availability as a so-called wildcard (property "definesWildcard"). Availability is the super class of concepts of entities of an academic program which can be defined in the course of the static modeling (like *course*, *examination*, *etc.*). The wildcard-reference defines which kind of individual results of a student can be assigned to the corresponding process step. Each result of a student therefore references an instance of Availability, too. In order to be able to assign an individual result with a certain process step,

this referenced instance of the result must be of the same specialization of Availability as the instance that is referenced by the process step as a wildcard. In addition, it must reference at least the same object- and datatype properties. Additional referenced properties are allowed, too. With this methodology, it is easy to model that an entity of the academic program only has to have, *e. g.*, a certain title: A process step has to reference an instance of a specialization of Availability as wildcard. In addition, this instance only has to reference a certain title. Then, it is defined that every result that references an entity of the same concept as the wildcard can be used for this process step if this entity references at least this certain title, too. A process step can be interpreted as successfully passed or not if a result is assigned to it that references an element that fits to the wildcard as described above. It has to be interpreted as successfully passed if the grade of the result fits to its grade scale rules (see [4]). In addition, each process step and each result can reference a couple of instances of specializations of Annotation. These specializations are also part of the flexible extension of the meta-model of the CMO. For example, *date* and *term* can be defined and then be used in connection with conditions to model rules like "in the average period of a study, an additional try is allowed". The classes Availability and Annotation can be arbitrarily extended in the course of the static modeling of academic programs.

A central part of the dynamic representation of academic programs is the arrangement of process elements of a process building an order. Doing this, possible sequences and rules that describe pre- and postconditions for taking these process steps can be modeled. But some major aspects can be modeled only intricately without modular composition of processes (see requirements in [5]). These are aspects like rules to retry a step, the inner part of a step (like the inner part of a module), minor subjects, or rules to calculate the grade of the final degree. In order to be able to model these dynamic aspects of academic programs smartly, the CMO defines the concept of internal processes. An internal process is a process that can be used instead of a couple of process steps. If a process step references such a specific internal process, it is possible to replace it with the internal process instead of assigning an individual result with it.

As a normal process — introduced in figure 1 — an internal process has a couple of process elements with a specific order. In addition, it has a process step that is used as a so-called "pattern" (property "isPattern"). While replacing a concrete process step with an internal process, the framework replaces this step with the pattern. Using a connector concept, the pattern is connected with other steps of the internal process in order to map the wildcard-reference of the replaced step to specific steps of the internal process (direction: pattern $\rightarrow$ steps). In the other direction, it is used to map a result (and, *e. g.*, a calculated grade) to the result of the replaced step (direction: pattern-result $\leftarrow$ results of the steps). By connecting the pattern instead of directly connecting the concrete replaced step, a single instance of an internal process can be used in order to replace a number of process steps and function as a template.

3 Distribution

To explain how this approach can be enriched applying it distributed, we take an examination office as an example. At academic institutions, certain information is often controlled at different places (see [1]). On the one hand, it is because academic institutions have a distributed structure. On the other hand, it is because of privacy aspects (see [6]). In many institutions, the individual results of the students, *e. g.*, are only stored at the examination offices, the supply of courses of the universities is often controlled by their faculties, and so on. Often, the structure of data is not compatible among each other, or there exists no integration. Thus, the employees of the examination offices might have to check manually if a result of a student is creditable for his curriculum or if the student is able to take certain examinations. Our approach supports the distributed structure and decision making of academic institutions: OWL and the URI-concept allow applications to access unique concepts and instantiations worldwide — even if they are stored distributed. The CMO meta-model is available under http://www-is.informatik.uni-oldenburg.de/eustel/cmo.owl. Each academic institution can extend the meta-model of the CMO (or an extension of it, created by, *e. g.*, an appropriate ministry of education). It can stand for a basic static representation of its academic programs (like basic definitions of, *e. g.*, *modules*). Finally, each institution can publish it on its website. Of course, it is also possible to use a common static representation or to extend an existing one. Each of the faculties of the institutions, then, might extend such static representation, instantiate it, and publish these dynamic representations of its academic programs on the web. In addition, each faculty can create and publish ontologies containing its supply of courses. For this purpose, they have to instantiate the static representation, too. These ontologies which contain the supply of courses, *e. g.*, can be used by web applications that generate semantic linked web pages containing the calendar of events[1]. On the same basis, the examination office can create private ontologies containing the status of each student. The entities of these ontologies can reference entities of the dynamic representation of the students' academic program and the exact actual entities for that the respective student has performed results. Using the model-interpreting framework, the individual possibilities of each student to continue his studies can be determined automatically by software of the examination offices. Keeping the student ontologies private, the examination offices can offer, *e. g.*, web services that allow the implementation and connection of service applications like learning management systems that offer individual curricula planning. Another very interesting aspect is the option to model possibilities for students to integrate results/courses that they have got at other academic institutions into curricula of programs of their own institution in a very simple way. It is required that both institutions use an extension of the CMO that stands for the static representation of their academic programs. These static representations do not have to be the same. Then, it

[1] An examplary implementation can be found at:
http://pixedia.de:8080/semaver/Show_Modules.do

is very easy to extend the dynamic representation of an academic program by possibilities in taking entities of the program of the other institution. To do so, one easy way is to simply add a process step that references an entity of the other institution and define that a specific step can be replaced by this new one via an internal process.

4 Evaluation and Outlook

In order to evaluate the CMO, we have on the one hand modeled concrete academic programs with this ontological concept. On the other hand, we have implemented the framework that is able to execute and to control the model interpreting process. Our implementation uses the JENA-Framework in order to reuse its ontological capabilities. To perform the model interpretation, a set of individual results of a student has to be incrementally mapped with the process steps of the modeled academic program. Doing this, the framework can detect for each of these interpretation steps which process steps of the academic program can be assigned with which individual results. If there are multiple possibilities to do such assignment, the decision can be made manually or automatically — depending on the kind of software that uses the framework. A planning assistance tool based upon our approach that uses the framework is also under development. It can help students that want to plan their individual curricula.

References

1. Schmees, M., Appelrath, H.J., Boles, D., Kleinefeld, N.: Erweiterung eines LMS um hochschultypische Softwaresysteme. In: M. Gaedke, R. Borgeest (eds.): Integriertes Informationsmanagement an Hochschulen: Quo vadis Universität 2.0?, Tagungsband zum Workshop IIM 2007, Universitätsverlag Karlsruhe, Karlsruhe, pp. 111–127 (2007)
2. Hackelbusch, R.: Handling Heterogeneous Academic Curricula. In: Min Tjoa, A., Wagner, R.R. (eds.) Proceedings of the Seventeenth International Conference on Databases and Expert Systems Applications (DEXA 2006), September 4-8, 2006, pp. 344–348. IEEE Computer Society Press, Los Alamitos (2006)
3. Gumhold, M., Weber, M.: Internetbasierte Studienassistenz am Beispiel von SASy. In: doIT Software-Forschungstag, Stuttgart, Fraunhofer IRB Verlag (2003)
4. Hackelbusch, R.: CMO – An Ontological Framework for Academic Programs and Examination Regulations. In: Jaakkola, H., Kiyoki, Y., Tokuda, T. (eds.) Information Modelling and Knowledge Bases XIX – Frontiers in Artificial Intelligence and Applications, vol. 166, pp. 114–133. IOS Press, Amsterdam (2008)
5. Tattersall, C., Janssen, J., van den Berg, B., Koper, R.: Using IMS Learning Design to Model Curricula. Interactive Learning Environments 15(2), 181–189 (2007)
6. Witt, B.C.: Datenschutz an Hochschulen. LegArtis Verlag (2004)