

An ontology-based approach to intelligent Instructional Design support

Helmut Meisel¹, Ernesto Compatangelo¹, and Andreas Hörfurter²

¹ Department of Computing Science, University of Aberdeen, AB24 3UE Scotland

² Kompetenzwerk, Riedering, Germany

Abstract We propose a framework for an Intelligent Tutoring System to support instructors in the design of a training. This requires the partial capture of Instructional Design theories, which define processes for the creation of a training and outline methods of instruction. Our architecture is based on a knowledge representation that is based on the use of ontologies. Reasoning based on Description Logics supports the modelling of knowledge, the retrieval of suitable teaching methods, and the validation of a training. A small exemplary ontology is used to demonstrate the kind of Instructional Design knowledge that can be captured in our approach.

1 Background and Motivation

Instructional design (ID) theories support instructors in designing a training. These theories describe processes for the creation of a training, outlining methods of instruction together with situations in which those methods should be used [1]. The design of a training involves the definition and the classification of learning goals, the selection of suitable teaching methods and their assembly into a course [2]. The learning outcome of a training is improved if these theories are applied in practice.

This paper proposes an architecture for an Intelligent Tutoring System that promotes the application of ID theories in practice. Such a system should assist an instructor in designing a training while educating him/her in Instructional Design. More specifically, this system should fulfill the following requirements:

- Assist its user in the selection of appropriate teaching methods for a training and encourage the application of a wide range of available teaching methods.
- Instruct its user about particular Teaching Methods (TMs)
- Highlight errors in the design of a training.

In our envisaged system (see Figure 1 for an architectural overview) ID experts maintain the system's knowledge directly. This requires a conceptual representation of ID knowledge that can be understood by non computing experts. Inferences are necessary to perform the validation and verification tasks as well as the retrieval of suitable teaching methods. We propose ontologies as the knowledge representation mechanism and show how set theoretic reasoning with ontologies can provide the necessary inferences.

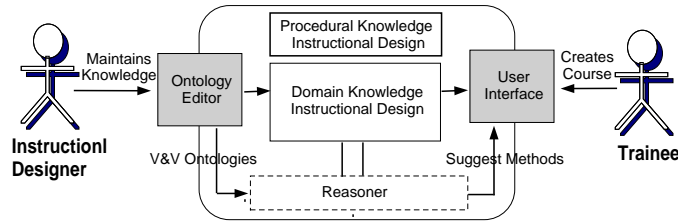


Figure 1. Architectural Overview

Related work in Artificial Intelligence has contributed to Instructional Design with tools targeting its authoring [3] and with systems for the (partial) automation of ID processes [4]. Early systems were built with heuristic knowledge such as condition-action rules [5]. A recent example of a rule-based approach is the Instructional Material Description Language (IMDL), that provides specifications for the automated generation of alternative course designs. IMDL considers instructional elements (i.e. learners or learning objectives) as pre-conditions and didactic elements (i.e. software components for the display of courses) as post-conditions. An inference engine creates alternative courses based on these specifications [6]. While this approach performs reasoning about ID knowledge, it does not provide a solution for an Intelligent Tutoring System. Rule-based systems create a “conceptual gap” between an expert’s knowledge and its representation in the rule-base [5]. Rule-based systems do not highlight relationships between corresponding rules and are therefore difficult to create or to maintain.

In e-learning, Meta-Data (such as the Learning Object Metadata) enable sharing and reuse of educational resources. However, current standards offer only limited support to describe pedagogical knowledge [7,8]. Our framework subsumes these standards, as Metadata classes can be viewed as ontology concepts. Ontologies are increasingly used to organize ID knowledge in e-learning platforms. In most cases, they facilitate the retrieval of semantically marked-up educational resources [9,10]. The Educational Modelling Language (EML) aims to provide a framework for the conceptual modelling of learning environments. It consists of four extendable top level ontologies that describe (i) theories about learning and instruction, (ii) units of study, (iii) domains, and (iv) how learners learn. EML ontologies could be reused in our framework. However, we aim to deploy the ontologies directly in an Intelligent Tutoring System rather than using them for design purposes only.

Knowledge representation and reasoning is equally important to meet the requirements for an Intelligent Tutoring System in ID. While rule-based systems could provide the necessary inferences, representational issues prevent their usage. Ontologies, on the other hand, are suitable means to capture ID knowledge. However, there is no standard inference mechanism for reasoning with ontologies.

2 Representation of ID knowledge

Instructional Design knowledge consists of a domain part about “methods of instructions” and their application and a procedural part about ID processes. Procedural knowledge of ID processes is of static nature and can be encoded in the User Interface as sequences of screens. Our framework mainly addresses the declarative part (i.e. the domain knowledge) of instructional design. The architecture of the proposed system relies on ontologies to capture this knowledge. Ontologies are “shared specifications of conceptualizations” [11] and encourage collaborative development by different experts. They capture knowledge at the conceptual level, thus enabling ID experts to directly manipulate it without the involvement of a knowledge engineer. Trainers may explore the ontology either by browsing through its entries or by querying it. Queries can also be assembled by the user interface in accordance with the information provided during the design process.

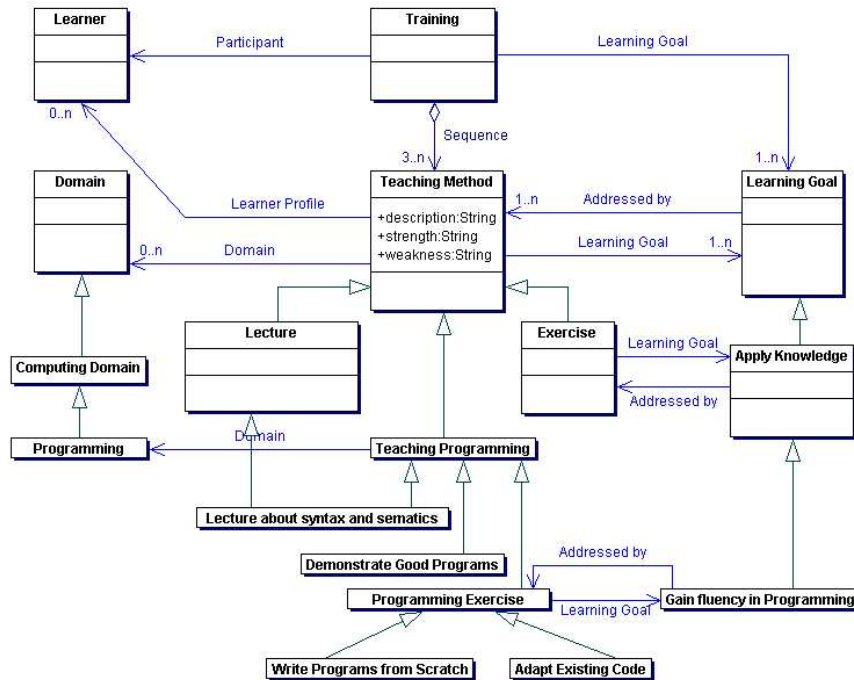


Figure 2. Exemplary Ontology

Ontologies represent knowledge in taxonomies, where more specific concepts inherit the properties of those concepts they specialize. This allows knowledge

reuse when an ontology needs to be extended. Figure 2 shows a small exemplary ontology to demonstrate, how ID knowledge can be captured. The part about **Teaching Programming** has been extracted from an existing evaluation of TMs in Computing Science [12]). Teaching methods are represented in the ontology by describing the situations they can be applied to (e.g. learning goals, characteristics of the learners, course domain, etc.). Instructions about the application of a teaching method can also be added to the ontology. In our exemplary ontology a **Training** is defined as a sequence of at least three different **Teaching Methods** - thus asserting that diverse teaching methods should be applied during a training. Moreover, each **Training** must have at least one **Learning Goal**. A **Learning Goal** is addressed by one or more **Teaching Method**. Note that it is thus possible to apply more than one teaching method to a learning goal. The selection of a **Teaching Method** in our example depends on (i) the **Domain** it is applied to (e.g. programming), (ii) the supported **Learning Goal** (e.g. application of a programming language) and (iii) the **Learner**. In our exemplary ontology, **Teaching Programming** is understood to be the set of all teaching methods that are applicable to **Programming**. A specific rule like *“It is generally agreed that fluency in a new programming language can only be attained by actually programming”* [12] can also be included in the specification. This rule is represented by linking the concepts **Programming Exercise** and **Gain Fluency in Programming** with the attributes *Learning Goal* and *Addressed by*.

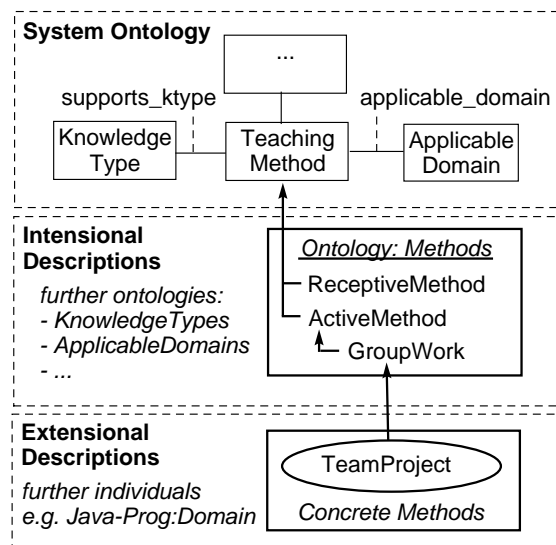


Figure 3. Layered knowledge representation

Instructional information about concepts must also be included in the ontology. This kind of information can be reviewed by a user in order to learn more about a particular teaching method. In the example, *description*, *strength*, and *weakness* of **Teaching Method** carry this information. Subclasses inherit this information from its superclasses. For instance, as **Lecture about syntax and semantics** is a subclass of both **Lecture** and **Teaching Programming**, it inherits the instructional information from both of them. A further benefit of multiple inheritance is that multiple views on the ontology can be defined. For instance, **Teaching Methods** can be classified according to learning strategy (e.g. Collaborative Learning, Discovery Learning, or Problem-based Learning).

Our architecture represents knowledge in three layers (see Figure 3):

- *System Ontology*. It defines categories of links between concepts (i.e. attributes). Links either point to instructional information about a TM or represent constraints for the selection of a TM. Only links representing constraints are subject to reasoning. Furthermore, the System Ontology specifies top-level concepts (e.g. Teaching Method, Learner, Learning Goal) which are relevant to model the specific implementation.
- *Intensional Descriptions*. Every class in the System Ontology is specialized by subclasses (e.g. teaching methods can be classified as either receptive or active methods). This level defines generic building blocks for the description of concrete teaching methods in the following layer.
- *Extensional Descriptions*. Individuals populate the ontologies defined in the previous layers. For instance, concrete teaching methods are elements of intensional descriptions (e.g. a Concrete Teaching Method “Writing a bubble sort program in Java” is an instance of the class **Programming Exercise**). The usage of individuals is essential for validation purposes as the reasoner can identify whether an individual commits to the structure defined in the ontology. This will be explored in the next section.

3 Reasoning services

We have developed an Intelligent Knowledge Modelling Environment called **ConcepTool** [13], which is based on a conceptual model that can be emulated in a Description Logic [14]. This enables deductions based on set-theoretic reasoning, where concepts are considered as set specifications and individuals are considered as set elements. Computation of set containment (i.e. whether a set A is included in a set B), inconsistency (i.e. whether a set is necessarily empty), and set membership (i.e. whether an individual x is an element of the set A) enables the introduction of the following reasoning services.

- *Ontology Verification and Validation*. Automated reasoning can provide support to ID experts during ontology creation and maintenance by detecting errors and by highlighting additional hierarchical links. For instance, if the ontology states that a TM must mention its strengths and weaknesses, the system will not accept TMs without this description. If the reasoner derives

a set containment relationship between two classes (e.g. a TM applicable to any domain and another TM applicable to the Computing domain only), it suggests the explicit introduction of a subclass link. In this case, all the properties of the superclass will be propagated to the subclass.

- *Retrieval of suitable teaching methods.* Teaching Methods are returned as the result of a query, stated as a concept description, which is issued to the reasoner. This retrieves all the individual TMs that are members of this concept. In our exemplary ontology, a query that searches all the TMs for the Computing Domain with the Learning Goal Gain fluency in Programming returns all the elements of the classes Programming Exercise, Write Programs from Scratch, and Adapt Existing Code. Note that individuals are not included in the exemplary ontology shown in Figure 2. The query concept can be generated by the user interface during the creation of a training. As reasoning in Description Logics is sound, only those teaching methods are suggested which satisfy the constraints.
- *Validation and Verification of a training.* Errors in the design of a training can be detected in two ways. The first way is to check whether a training commits to the axioms of the ontology. The training to be validated is considered as an instance of the Training class and thus needs to commit to its structure. Possible violations w.r.t. our exemplary ontology might be (i) to define only two Teaching Methods (whereas at least three are required), (ii) forget to specify a Learning Goal, or (iii) forget to address a learning goal with a Teaching Method. A further possibility to validate a training is to define classes of common errors (e.g. a training with receptive teaching methods only) and check whether the training under validation is an instance of a common error class.

4 Discussion and future research

This paper presents an architecture for an Intelligent Tutoring System in Instructional Design (ID). Such an architecture addresses declarative ID knowledge, which can be directly manipulated by ID experts as this knowledge is represented explicitly using ontologies. Automated reasoning fulfills the requirements stated in Section 1 (i.e. assist users in the selection of appropriate teaching methods and highlight errors in the design of a training). The requirement to instruct a user about particular teaching methods can be achieved by attaching instructional information to each TM.

Our framework does not commit to any particular ID theory. In principle, it can be applied to any approach as long as this allows the construction of an ontology. However, as the framework does not include a representation of procedural knowledge, the user interface (most likely) needs to be developed from scratch. Nevertheless, we anticipate that changes of an ID process will rarely occur.

As the CONCEPTOOL ontology editor is reasonably complete, we aim to develop ontologies and implement the User Interface part of this architecture.

This will investigate the validity of the assumption that a user who is not an expert in knowledge modelling can represent ontologies without the help of a knowledge engineer.

Acknowledgements

The work is partially supported by the EPSRC under grants GR/R10127/01 and GR/N15764.

References

1. C., R.: Instructional design theories and models: A new paradigm of instructional theory. Volume 2. Lawrence Erlbaum Associates (1999)
2. Van Merrinboer, J.: Training complex cognitive skills: A four-component instructional design model for technical training. Educational Technology Publications (1997) ISBN: 0877782989.
3. Murray, T.: Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education* **10** (1999) 98 – 129
4. Kasowitz, A.: Tools for automating instructional design. Technical Report EDO-IR-98-1, Education Resources Information Center on Information Technology (1998) <http://ericit.org/digests/EDO-IR-1998-01.shtml>.
5. Mizoguchi, R., Bourdeau, J.: Using ontological engineering to overcome common ai-ed problems. *International Journal of Artificial Intelligence in Education* **11** (2000) 107–121
6. Gaede, B., H., S.: A generator and a meta specification language for courseware. In: Proc. of the World Conf. on Educational Multimedia, Hypermedia and Telecommunications 2001(1). (2001) 533–540
7. Pawlowski, J.: Reusable models of pedagogical concepts - a framework for pedagogical and content design. In: Proc. of ED-MEDIA 2002, World Conference on Educational Multimedia, Hypermedia and Telecommunications. (2002)
8. M., R., Wiley, D.: A non-authoritative educational metadata ontology for filtering and recommending learning objects. *Interactive Learning Environments* **9** (2001) 255–271
9. Leidig, T.: L3 - towards an open learning environment. *ACM Journal of Educational Resources in Computing* **1** (2001)
10. Allert, H., et al.: Instructional models and scenarios for an open learning repository - instructional design and metadata. In: Proc. of E-Learn 2002: World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education. (2002)
11. Uschold, M.: Knowledge level modelling: concepts and terminology. *The Knowledge Engineering Review* **13** (1998) 5–29
12. Nicholson, A.E., Fraser, K.M.: Methodologies for teaching new programming languages: a case study teaching lisp. In: Proc. of the 2nd Australasian conference on Computer science education. (1997)
13. Compatangelo, E., Meisel, H.: \mathcal{K} -ShaRe: an architecture for sharing heterogeneous conceptualisations. In: Proc. of the 6th Intl. Conf. on Knowledge-Based Intelligent Information & Engineering Systems (KES'2002), IOS Press (2002) 1439–1443
14. Horrocks, I.: FaCT and iFaCT. In: Proc. of the Intl. Workshop on Description Logics (DL'99). (1999) 133–135