Constraint-Based Validation of Adaptive e-Learning Courseware

Mark Melia, *Student Member*, *IEEE*, and Claus Pahl, *Member*, *IEEE*

Abstract—Personalized e-learning allows the course creator to create courseware that dynamically adapts to the needs of individual learners or learner groupings. This dynamic nature of adaptive courseware makes it difficult to evaluate what the delivery time courseware will be for the learner. The course creator may attempt to validate adaptive courseware through dummy runs, but cannot eliminate the risk of pedagogical problems due to adaptive courseware's inherent variability. Courseware validation checks whether adaptive courseware conforms to a set of pedagogical and nonpedagogical requirements. The validation of adaptive courseware limits the risk of pedagogical problems at delivery time. In this paper, we present our approach to adaptive courseware validation using the Courseware Authoring Validation Information Architecture (CAVIAr). We outline how CAVIAr captures adaptive courseware authoring concerns and validates courseware using a constraint-based approach. We also describe how CAVIAr can be integrated with the state of the art in adaptive e-learning and evaluate our validation approach.

Index Terms—Adaptive, courseware, validation, CAVIAr, modeling, interoperability.

1 Introduction

Adaptive or personalized courseware is courseware which moves away from the "one size fits all" paradigm of traditional courseware and instead looks to adapt to the individual needs of learners in terms of their learning goals, knowledge of a subject, and learning style, among other things.

The construction of adaptive or personalized courseware is a "complex, time-consuming and expensive task" [1]. Recent advances in dedicated tools, such as My Online Teacher (MOT) [2] and the Adaptive Courseware Construction Toolkit (ACCT) [1], have raised the level of abstraction that the course creator works at when creating adaptive courseware. These tools go some way toward providing a more intuitive interface for designing adaptive courseware, but do not provide a method for validating the adaptive courseware created. We propose validation to be a part of the adaptive courseware construction process, an activity that automatically checks newly constructed adaptive courseware for a range of problems. The types of problems that courseware validation detects include the incorrect sequencing of learning resources, an instructional design being applied incorrectly, or an inconsistency in the adaptive courseware structure.

Adaptive courseware validation allows the course creator to minimize the pedagogical problems that learners must deal with when using immature adaptive courseware. The literature notes the importance of postconstruction/predelivery course validation or "course auditing" as an essential part of a holistic course construction methodology [3], [4]. We believe this to be even more important with adaptive courseware due to the additional complexity in its design, making construction more complicated and harder to validate manually.

Courseware evaluation typically makes use of learner models in order to analyze learner interaction with the courseware [5]. This approach mirrors recognized formative and summative evaluations of traditional courses [6], [7]. Courseware validation is a predelivery activity and must therefore validate courseware without the learner model data that are generated at delivery time. Existing courseware validation efforts attempt to overcome this problem by simulating the learner's interaction with courseware [8], [9]. In this paper, we present a novel, constraint-based approach to courseware validation. The approach is based on software modeling technology that allows for the formal definition of courseware and its requirements. The course creator can then define a constraint-based validation model in terms of the courseware requirements that must be true for the newly constructed courseware. Our approach is driven by the validation needs of adaptive courseware by allowing the explicit definition of adaptation concerns such as the anticipated learner models. Furthermore, our approach is not affected by the complexity that adaptive decision points introduce to personalized courseware.

To allow for the validation of adaptive courseware, in Section 3, we define the Courseware Authoring Validation Information Architecture (CAVIAr) using the Meta Object Facility (MOF), an abstract language for defining modeling languages such as the Unified Modeling Language (UML) [10]. The CAVIAr abstract syntax can be constrained using the Object Constraint Language (OCL). OCL is an OMG-defined constraints language used to constrain MOF-based models. It is based on set theory and logic [11]. In Section 4, we outline the types of constraints that can be imposed on the CAVIAr metamodels and how these constraints can be used to validate adaptive courseware. In Section 5, we

• *The authors are with the School of Computing, Dublin City University, Dublin 9, Ireland. E-mail: {mmelia, cpahl}@computing.dcu.ie.*

Manuscript received 15 Sept. 2008; revised 27 Nov. 2008; accepted 9 Jan. 2009; published online 14 Jan. 2009.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLTSI-2008-09-0082.
Digital Object Identifier no. 10.1109/TLT.2009.7.

demonstrate how CAVIAr courseware validation can be integrated into existing courseware authoring tools. We evaluate our approach to adaptive courseware validation in Section 6, followed by a critical analysis of the state of the art in courseware validation in Section 7. Our concluding remarks are presented in Section 8.

We shall begin our discussion looking at the state of the art in constructing adaptive e-learning courseware and motivate the need for courseware validation.

2 Constructing Adaptive E-Learning Courseware

The widely accepted ADDIE model is a general-purpose model for constructing courseware consisting of five phases, **A**nalysis, **D**esign, **D**evelopment, **I**mplementation, and **E**valuation. Courseware validation fits into the ADDIE model as a preimplementation (predelivery of the learning material to the learner), postdevelopment activity.

Constructing adaptive courseware is not a trivial task, it is a complex task which involves a variety of skills. In constructing adaptive courseware, the course creator (or team of course creators) must define how a given courseware will adapt to aspects of learning, such as an evolving learner model, a subject domain, and/or an instructional strategy in use. In general, the definition of adaptive courseware involves defining where the course flow branches based on some variable, for example, the learner's knowledge. These courseware flow branches are defined in the courseware adaptivity logic. Defining the adaptivity at the courseware level is generally done in a programming language like syntax [1], [12], [13]. Adaptive courseware authoring tools provide an intuitive user interface for the course creator to define adaptivity, by raising the level of abstraction the course creator works at.

The MOT system [14], developed at Eindhoven University of Technology, allows course creators to create adaptive courses in Adaptive Educational Hypermedia (AEH) using the LAOS system of layered models [15]. The LAOS model has two types of models—static and dynamic. The static model describes the domain, and course-oriented goals and constraints on a domain model. The dynamic model defines adaptive rules using a programming language like syntax. The adaptive rules define the course flow branches and are defined using data from the LAOS static models. As the MOT system is solely an authoring system, the adaptive courseware must be exported into an AEH delivery system such as AHA! [16] or WHURLE [17].

The ACCT allows the course creator to design adaptive courseware based on the principle of separating the "key design elements of personalized e-learning" [1]. Adaptivity is provided through the combination of models, each of which addresses a key design element. To make the definition of an adaptive courseware more intuitive, the ACCT uses abstraction by providing design patterns for courseware structure known as "Narrative Structures." Adaptive logic is defined in "Narrative Attributes," which define how a specific learning resource is selected to be delivered to an individual learner.

The IMS Learning Design (LD) specification allows for the definition of adaptive courseware, by defining learning flow conditions on learning activities [18]. Constructing courseware defined using IMS LD requires the course creator to have an in-depth knowledge of the specification. The COLLAGE tool allows the course creator to define IMS LD adaptive behavior by raising the course creator level of abstraction [19]. This is done by providing a catalogue of Computer Supported Collaborative Learning (CSCL) design patterns that are implemented in IMS LD.

Adaptive courseware authoring tools raise the level of abstraction that the course creator works at. Due to this, and the level of complexity involved in defining adaptive strategies through rules, the course creator may wish to check that courseware to be delivered to learners is as envisioned and required. The course creator may wish to validate aspects of the courseware, such as:

- All topics covered in the courseware adapt to the needs of every anticipated learner stereotype; for example, learners with particular learning styles will be directed to learning material that suits their style.
- The courseware adapts in some sort of standardized way for all topics or for selected topics; for example, when a learner has a knowledge level below a specific threshold for compulsory topics, the learner is directed to supplementary material.
- All topics in the courseware have a lecture explaining the topic's central concept.
- Topics in the courseware do not go over a certain time limit regardless of how the courseware adapts.

Although some simple aspects of adaptive courseware design can be manually checked by the course creator, such as the type of Learning Objects (LOs) in a courseware topic, it is time consuming and must be checked for each possible permutation of learning paths through courseware. Some more complex aspects of adaptive courseware design, such as ensuring that the courseware adapts in some standardized way, are much more difficult, if not impossible, to check manually.

3 CAVIAr Courseware Construction Concern (C4) Models

The CAVIAr consists of a set of data models and a validation model. The data models are used to capture the courseware construction concerns used to define and develop courseware [20]. This set of data models is known as the CAVIAr Courseware Construction Concern (C4) Models. The C4 models are as follows:

- Domain model—models the subject domain to be covered by the courseware.
- Learning context model—consists of learner model representations and domain pedagogic information. This model is principally responsible for capturing adaptivity concerns that are defined as anticipated learner stereotypes in terms of the domain model.
- Learning resources model—defines the learning resources used in courseware.
- Courseware model—courseware constructed by the course creator.

Fig. 1 provides an overview of the CAVIAr and its constituent models, defined in UML. The root *CAVIAr*

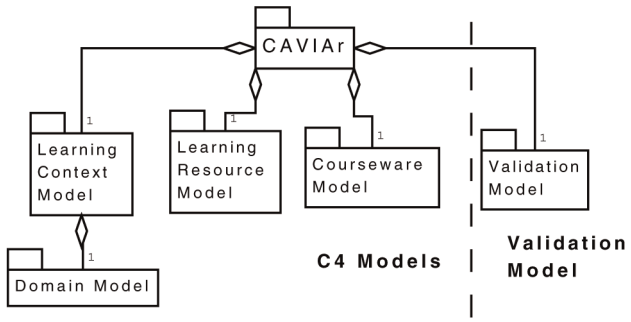

Fig. 1. Overview of the CAVIAr models.

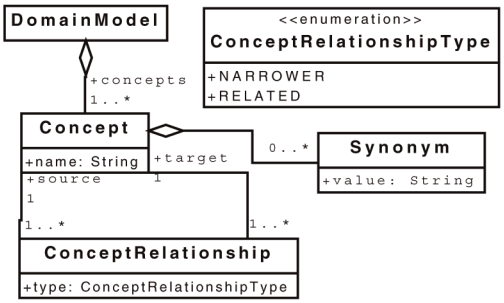

Fig. 2. Domain model abstract syntax defined in MOF.

modeling element consists of a learning context model, a learning resource model, a courseware model, and a validation model. The domain model is part of the learning context model.

The C4 models represent the courseware requirements in the form of a learning context model, and the actual courseware to be delivered to learners in the form of a courseware model. The validation model is defined by constraining the allowable courseware model definitions. To do this, a constraint language is used to define constraints on the courseware model abstract syntax definition. These constraints are based on requirements data provided in the learning context model. The validation model definition is covered in detail in Section 4.

In the following sections, each C4 model is introduced. This is done by first outlining the model's purpose, and then outlining the model in terms of its abstract syntax defined in MOF.

3.1 The Domain Model

The sole purpose of the domain model in CAVIAr is to express the curriculum knowledge structure, this being the knowledge the courseware covers. It is worth noting that all knowledge in the domain model does not have to be covered by the courseware. The knowledge covered by the courseware may be a subset of the knowledge outlined in the domain model.

The CAVIAr domain model's metamodel is defined in MOF and is depicted in Fig. 2. The *concept* is the domain model primary building block. Concepts are related to other concepts by a *ConceptRelationship*. A concept relationship has a direction from its *source* concept to its *target* concept. There are two types of concept relationships derived from the SKOS concept semantic relationships [21]. These are defined in the enumeration *ConceptRelationshipType*:

- *NARROWER*—the source concept has a broader semantic scope than the target concept. This allows for a taxonomic relationship between concepts.
- *RELATED*—there is a semantic relationship between two concepts. This relationship is symmetric.

3.2 The Learning Context Model

The learning context model defines domain pedagogic information in terms of the domain model. The learning context model is made up of:

- conceptual instructional constraints and
- learner stereotype definitions.

The conceptual instructional constraints specify sequencing constraints, where knowledge of one concept is necessary to understand another (e.g., addition must be taught before multiplication).

A learner stereotype definition allows the course creator to define learner groupings in terms of their learning goals and assumed initial knowledge prior to starting the courseware. The course goal and assumed initial knowledge are defined as knowledge elements in terms of the domain model. Knowledge elements are defined by a knowledge level, a knowledge type, and a domain model concept.

In Fig. 3, we outline the learning context model's metamodel in MOF. One of the central elements of the learning context is the *Concept* from the domain model, as the learning context is defined on the domain model.

An additional relationship is defined on the domain model, *ConceptPreReq*. This relationship allows the course creator to define a prerequisite sequencing constraint between two concepts. A *ConceptPreReq* relationship defines a relationship between a concept and a *KnowledgeElement*, where the prerequisite is satisfied when a learner has obtained the knowledge defined by the *KnowledgeElement*. A *KnowledgeElement* is defined with an attribute for the knowledge type and the knowledge level, and references a concept in the domain model.

The learning context model defines a learner stereotype in terms of goals and presumed knowledge. The goals describe a learner stereotype's desired knowledge state and presumed knowledge describes the expected knowledge of the learner stereotype prior to taking the courseware. The *Goal* and *PresumedKnowledge* model elements are types of *KnowledgeConstraint*.

A central aspect to the learning context definition is the *KnowledgeElement*, which is composed of a reference to a concept, a knowledge type, and a knowledge level. Knowledge types are defined as either "verbal information" or "intellectual skills," which are the two learning outcomes, defined by Gagné et al. [22], that can be defined in terms of domain model concepts. The knowledge level allows the course creator to define the level of knowledge a learner stereotype has in the associated concept, between 0 and 1, where 1 is expert knowledge akin to scoring full marks in a test on the associated concept and 0 indicating no knowledge. The knowledge level indicates a level of a particular type of knowledge.

A learner stereotype ls_i is defined by a name and a set of *KnowledgeConstraints*. The name is the stereotype

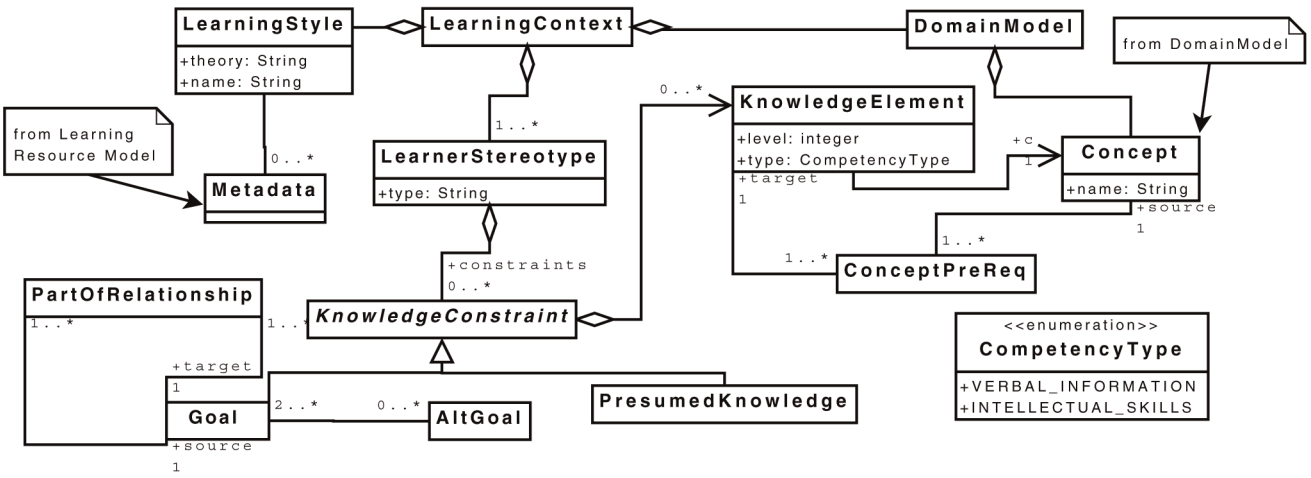

Fig. 3. Learning context model abstract syntax.

identifier. There are two types of knowledge constraints that can be defined on a learner stereotype, *Goal* and *PresumedKnowledge*. *PresumedKnowledge* defines the knowledge elements that the course creator believes the learner stereotype to have prior to the initiation of the courseware. The *Goal* knowledge constraints is a representation of the knowledge state in terms of knowledge elements that the learner must have after taking the courseware. The *PartOfRelationship* allows for the construction of composite goals. If a *Goal* g1 is part of another *Goal* g2, the *LearnerStereotype* has an overall goal that is a union of the knowledge elements in the two goal sets $g1 \cup g2$. Alternative goals are defined using the *AltGoal* construct, where an *AltGoal* instance is associated with goals that are alternatives of each other.

To illustrate how a learning context model is defined, we have outlined an example in Fig. 4. In this example, a learning context model is defined on a simple databases domain model (in gray) that has two learner stereotypes defined: *CS_Students* and *General_Students*. After taking the courseware, all learners must understand ER_Modeling. CS_Students must also understand either relational algebra or relational calculus. We assume that all learners have excellent knowledge on information systems. As can be seen in the diagram, *KnowledgeConstraints* (i.e., goals1-3 and presumedKnowledge1) link the stereotypes to concepts in the domain model. The *KnowledgeElement* knowledge level and type are captured in the relationship between a *KnowledgeConstraint* and domain model *Concept*.

The learning context model also allows the course creator to define the learning styles that he or she wants to use in validation. Learning styles can then be associated with LO metadata types. Capturing the anticipated learning styles that will be used in the courseware allows for the validation model to validate courseware in terms of how learning styles are accommodated. This is a similar approach to the integration of learning styles into adaptive strategies defined by Stash et al. [23].

3.3 The Learning Resource Model

The learning resource model contains representations for learning resources used in the courseware. The learning resource representation is based on the IEEE LOM standard [24], allowing for a direct mapping from LOM to the learning resource model.

In Fig. 5, we outline an excerpt of the learning resource model's metamodel in MOF. As outlined in the figure, a *LearningResourceModel* is composed of *Resources*. There are two types of *Resource*: *LO* and *Service*.

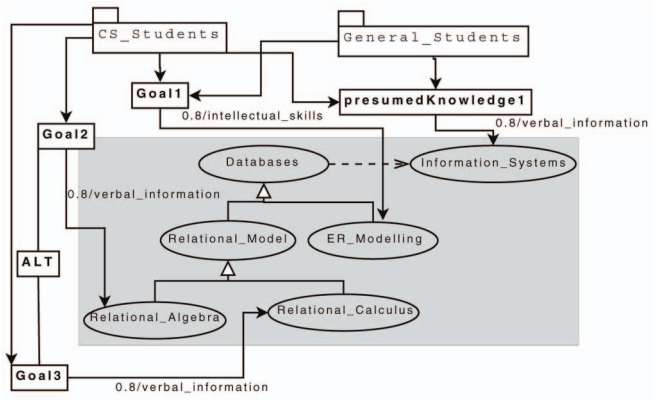

Fig. 4. Example CAVIAr learning context model with the domain model in gray.

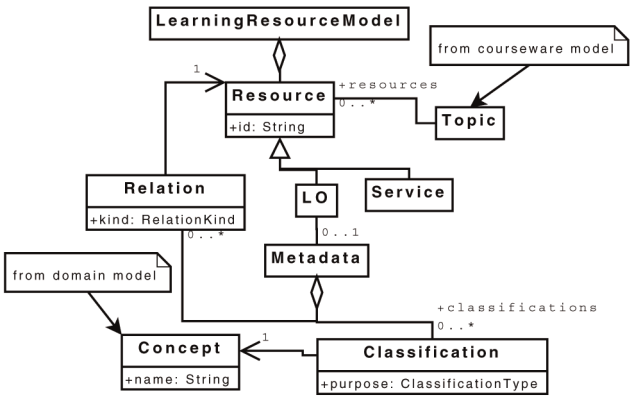

Fig. 5. Excerpt of learning resource model abstract syntax.

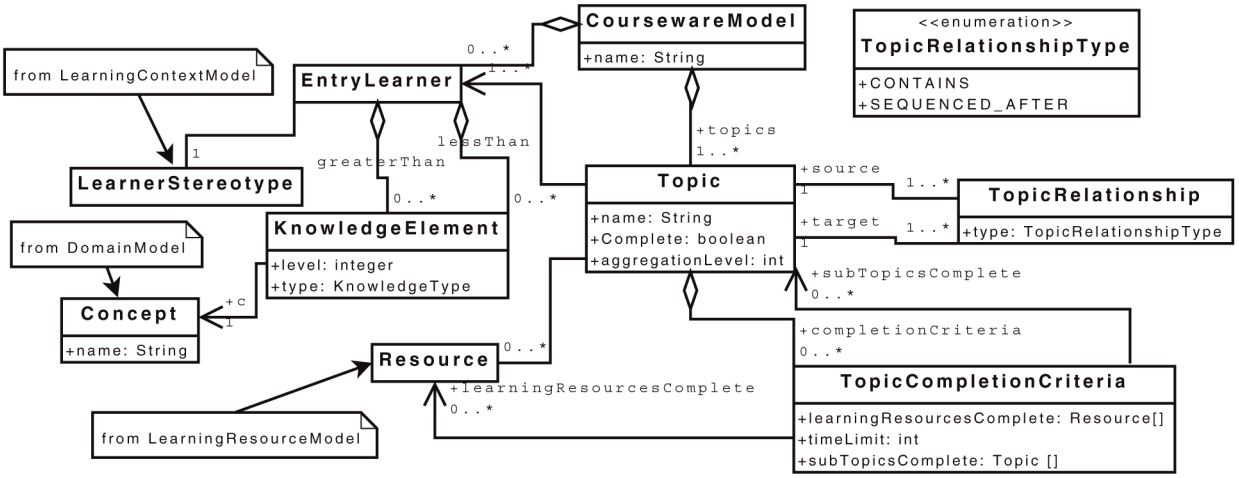

Fig. 6. Abstract syntax for courseware model defined using MOF.

The *LO* is associated with *Metadata*. The *Metadata* element is composed of five types of LO metadata: *Relation, Classification, General, Educational,* and *Technical*. All modeling elements contained in *Metadata* have attributes which are used to describe the associated *LO*. The *Classification* associates the *LO* with a domain model concept for some *purpose*, where purpose is an attribute of the *Classification* class. The *Relation* metadata relates a LO to other LOs.

3.4 The Courseware Model

The CAVIAr courseware model defines the structure, content, and behavior of courseware. The main specifications for defining courseware are the ADL SCORM 2004 [25] and IMS Learning Design [18]. Here, we outline a language for defining courseware that is independent of (although inspired by) these specifications. This allows for minimal disruption to CAVIAr-based courseware validation tools as the specifications mature.

Fig. 6 illustrates the courseware metamodel defined using MOF. Courseware is principally made up of *Topic* instances. A *Topic* has a *name* and an *aggregationLevel*, where a *name* uniquely identifies the topic and the *aggregationLevel* allows for various topic granularity levels (e.g., lesson, module). Topics are related to other topics via a *TopicRelationship*. A *TopicRelationship* has a type. The type is defined as a *TopicRelationshipType*. Topics can contain other topics through the *TopicRelationshipType—PART_OF*. Explicit topic sequencing definitions are specified using the *TopicRelationshipType—SEQUENCED_AFTER*. The *SEQUENCED_AFTER* relationship specifies when one topic must be covered before another Topic. Topics can reference zero to many learning *Resources*.

An *EntryLearner* is a condition, associated with a Topic. The condition must be true for a learner to enter the associated topic. An *EntryLearner* consists of one *LearnerStereotype* and zero to many *lessThan* or *greaterThan KnowledgeElements*. The *LearnerStereotype* defines a learner group that can access the topic. For example, only the learners in the "beginner" stereotype can access a given topic. *KnowledgeElements* allow for additional *EntryLearner* conditions to be defined based on a learner's knowledge. For example, there may be a topic that should only be covered when the learner has achieved expert "verbal information" knowledge in a related courseware concept.

The *TopicCompletionCriteria* allows the course creator to express conditions for when the *Topic* is deemed complete. For example, the course creator may only want the learner to complete two out of three of the contained topics, a *TopicCompletionCriteria* is expressed for each of the permutations of completion. The *TopicCompletionCriteria* also allows for a simple time limit on a *Topic*.

To illustrate how a courseware model can be defined, we have outlined a simple example in Fig. 7. In this courseware model, there are four topics (from database theory). *ER_Modeling* and *Relational_Modeling* topics are contained within the *Databases* topic. The topic *ER_Modeling_Help* is contained within the *ER_Modeling* topic and contains some supplementary material on ER modeling. The *ER_Modeling_Help* topic has an *EntryLearner* condition, where only *CS_Students* (computer science students) with verbal information knowledge in the concept *ER_Modeling* less than 0.2 may enter the topic. This means that a *CS_Student* learner struggling with ER Modeling will be delivered supplementary material on the topic. This demonstrates how simple adaptive behavior is defined in a courseware model.

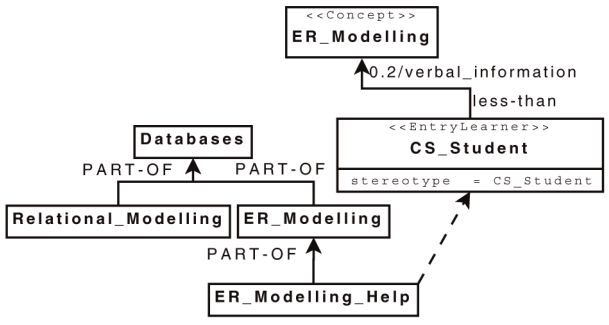

Fig. 7. Example CAVIAr courseware model depicting a databases courseware.

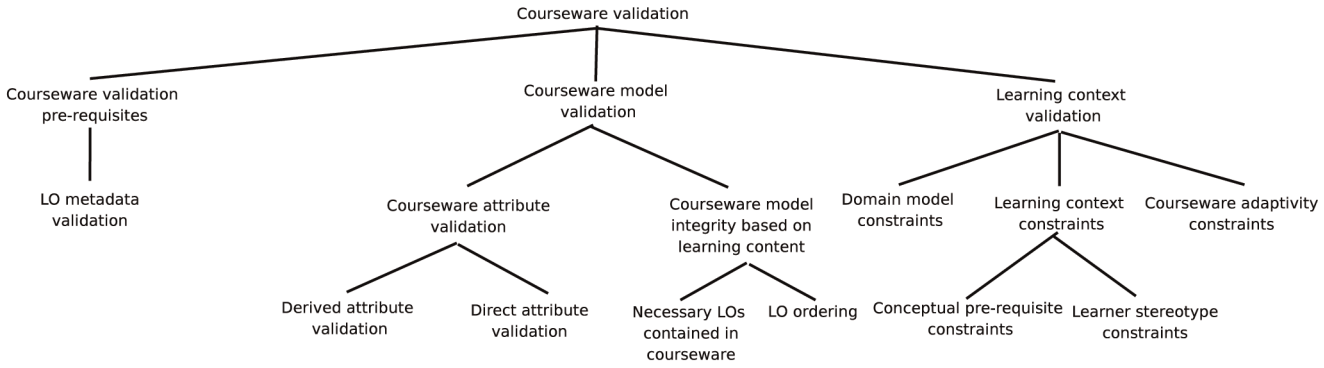

Fig. 8. Classification of CAVIAr validation constraints.

4 Adaptive Courseware Validation

Adaptive courseware validation works by defining constraints on the CAVIAr metamodels [26]. These constraints must then be adhered to in all instances of the CAVIAr metamodel—a CAVIAr model defined by the course creator. The course creator can therefore define constraints that must be true for an adaptive courseware model in terms of the CAVIAr metamodel.

In determining the CAVIAr validation model, we can split the types of validation constraints into three key categories (see Fig. 8):

- *Validation prerequisites.* This type of validation does not check for instructional features in the CAVIAr model, but checks that data needed for validation are available in the CAVIAr C4 models. The validation prerequisites allows the course creator greater confidence in validation. In the interest of space, we will not look at this category of validation in detail.
- *CAVIAr courseware model validation.* Validation based solely on the courseware model.
- *CAVIAr learning context validation.* The learning context model (Section 3.2) defines the adaptivity and courseware requirements. Using this type of validation, we can check these requirements.

To formally define courseware constraints and to allow for automated constraint checking, constraints are defined in OCL. OCL is quite a verbose language, designed for software engineers. To raise the level of abstraction when using OCL, we have defined several OCL functions that define common operations for defining CAVIAr constraints. These are denoted in capital letters. In defining OCL, we also assume that all data needed for validation are available, for example, that all LOs in the courseware have been sufficiently annotated with metadata.

This section concludes by illustrating how a validation model can be defined to ensure that courseware uses a particular instructional design.

4.1 Courseware Validation Based on the Courseware Model

Courseware model validation looks to validate the courseware model in isolation from the learning context. Validation based on the courseware model lends itself to two types of validation:

- *Courseware attribute validation.* This type of validation validates a courseware attribute against an externally defined value.
- *Courseware model integrity based on courseware learning content.* Validates courseware model, ensuring it is structured correctly for the learning content it contains.

4.1.1 Validating CAVIAr Attributes against External Values

This is the simplest type of validation the course creator can define. It involves comparing an attribute of the courseware model with an external value, or deriving an attribute value from a defined set of courseware attributes and comparing that to some external value. The external value is an alphanumeric value. The comparison tests the relation between that external value and the one from the courseware model using a relational operator.

An example case of this constraint is where the duration of each LO in courseware can be no more than 30 minutes. This type of rule might be used in an environment where the learner's time is an expensive resource. In order to evaluate this, the duration of each LO must be evaluated to be below the maximum specified time. The duration of the LO can be found in the LO's metadata, it is an attribute of a LO's technical metadata. In Listing 1, the invariant constrains this attribute to be less than 30 (minutes).

Listing 1. OCL rule that specifies LOs in CAVIAr cannot be longer than 30 minutes in duration.

```
context Metadata
    inv LO_violate_max_duration_time : technical.duration < 30
```

A variety of aggregated data can also be derived from the courseware model for validation. In Listing 2, the LO duration value is used to evaluate the duration of the courseware as a whole. This involves specifying how to evaluate the time of topics in courseware (*getTopicTime()*) and then specifying a courseware invariant which retrieves the sum of all topic times in the courseware and ensures it is less than the maximum courseware time.

Listing 2. OCL constraint that evaluates the courseware time from its contained LOs and specifies a maximum courseware time of 1,000 minutes.

```
context Topic
    def: getTopicTime(): Integer =
        self.resources->select(oclIsOfType(LO))
        ->iterate(i; res:Integer=0| res = res+i.metadata.technical.duration)
        +self.CONTAINED_TOPICS->iterate(j;a:Integer=0|a= a+j.getTopicTime())
context Courseware
    inv max_courseware_time_exceeded :
        self.topics->iterate(i;res:Integer=0|res=res+i.getTopicTime()) < 1000
```

4.1.2 Courseware Integrity Based on Courseware Learning Objects

We can check that the courseware use of LOs is correct through OCL. An example of such an integrity check is to ensure that if a LO in courseware references another LO, the referenced LO is also in the courseware.

A more complex integrity check would involve checking the sequencing of topics and ensuring that the sequencing of topics is correct for LOs in the courseware. For example, should the *relation* be of type *RelationKind.BASED_ON*, such that LO_b is BASED_ON LO_a, the instructional designer can ensure that LO_a is sequenced first. This is done by defining an OCL constraint which specifies there must be a SEQUENCED_AFTER topic relationship between the topic which contains LO_a and that of LO_b, stating that the topic containing LO_a is sequenced first.

OCL can also be used to validate that each of the learning styles in the learning context model is accommodated or used to verify that LOs which suit a particular learning style are only delivered to learners of that learning style. This is done by verifying that entry learner conditions exist on topics that contain LO types associated with a particular learning style.

4.2 Validating Courseware Using the CAVIAr Learning Context Model

In validating courseware using the CAVIAr learning context model, our aim is to ensure that the courseware covers the courseware requirements stated in the learning context model.

We define three types of instructional constraints using the learning context model:

- instructional constraints using the domain model only,
- instructional constraints using the overall learning context including the domain model and the learner sterotype information in CAVIAr, and
- courseware adaptivity constraints—check entry learner constraints on topics define the correct personalization strategy.

4.2.1 Domain Model Constraints

The courseware model is associated with a domain model through LO metadata. Each LO in the courseware is classified using domain model concepts. The association between courseware topics and domain model concepts can be used to examine the courseware model in the context of the domain model. Validation rules can be defined based on this association comparing the courseware model with the domain model. Here, we will look at how OCL can be used to compare a courseware model with its related domain model to determine the courseware model's validity.

Conceptual relationships in the domain model define how two concepts are related to each other. We can use this relationship to derive instructional design rules that can be validated against the courseware model. For example, the domain model's *NARROWER* concept relationship can be used to define a sequencing constraint between the courseware topics, which have LOs that are related via the *NARROWER* relationship. The *NARROWER* conceptual relationship is used in the instructional constraint defined in Listing 3, which specifies that all topics covering more specialized concepts, must be sequenced after topics covering broader domain model concepts.

Listing 3. OCL constraint which uses conceptual relationship semantics to define an instructional constraint where all topics covering broader concepts are sequenced before those more specialized.

```
context Topic
    inv sequencing_conforms_to_domain_model: self.TOPIC_CONCEPTS
            ->iterate(x:Concept; a:Set(Concept)=Set{}| a->union(x.NARROWER))
        -self.SEQUENCED_AFTER
            ->iterate(y:Topic; b:Set(Concept)=Set{}| b->union(y.referencedResource))
            ->select(oclIsTypeOf(LO))->asSet()
        = Set{}
```

4.2.2 Validating the Courseware Model Using the Learning Context Model

The learning context model allows us to define two types of constraints:

- *Conceptual prerequisite constraints.* Conceptual sequencing constraints on the courseware domain model, here we assume one common domain model for all LO conceptual annotation.
- *Learner stereotype constraints.* Can be used to ensure that the needs of the stereotype grouping are covered in the courseware.

We define OCL invariant constraints to check the constructed courseware for the conceptual prerequisite relationships specified in the CAVIAr learning context model.

The first invariant checks that, if there is a prerequisite relationship between two concepts where concept c_{pre} is the prerequisite of concept c, then c_{pre} will be covered in the courseware before concept c.

To define this, in OCL, the course creator can create two sets for each topic, the first containing all the prerequisite concepts of the concepts covered by $topic_i$, set P, and the second set containing the concepts that will definitely be covered prior to the learner getting to $topic_i$, set C. The second set is constructed by getting the concepts of the topics that are related to the $topic_i$ through the *SEQUENCED_AFTER* topic relationship and $topic_i$ is the source. The difference of these two sets is then sought. The difference of these two sets must be an empty set (i.e., there are no concepts that are prerequisite concepts and not covered by topics sequenced before the topic in question, $P - C = \emptyset$). We have outlined this OCL invariant in Listing 4.

Listing 4. OCL to ensure that prerequisite concepts are always sequenced before the topic that requires them.

```
context Topic
    inv conceptual_prerequisite_rule: self.TOPIC_CONCEPTS
        ->iterate(x:Concept; a:Set(Concept)=Set{}| a->union(x.PREREQUISITE)))
        -self.SEQUENCED_AFTER->iterate(y:Topic; b:Set(Concept)=Set{}|
            b->union(y.concepts))
        = Set{}
```

The constraint in Listing 4 allows us to check the sequencing of courseware is conceptually correct, but does not take into account a learner's prior knowledge. For example, if the set resulting from the difference operation resulted in one concept c_1, where $c_1 \in P$, i.e., in the prerequisite concept set, but is presumed knowledge of a particular learner stereotype taking the course. The courseware should still be deemed valid for that learner stereotype as the concept, although not covered prior to the topic that needs it, is knowledge the learner is assumed to already have, defined as set L.

In Listing 5, a generic approach to dealing with presumed knowledge is taken $(P - C) - L = \emptyset$. The conceptual prerequisite constraint violation is sought first as in Listing 4, then any violating concepts are compared against the common assumed knowledge for all the learners to take this courseware (i.e., union of all presumed knowledge for all learner stereotypes). Only when there are still outstanding concepts will the constraint *conceptual_prerequisite_rule*, in Listing 5, be flagged as invalid.

Listing 5. OCL ensuring conceptual prerequisites are sequenced before topics that require them and also that assumed learner knowledge is acknowledged.

```
context CAVIAr
    inv conceptual_prerequisite_rule: self.courseware.ALL_TOPICS
        ->forAll(
            (TOPIC_CONCEPTS
                ->iterate(x:Concept; a:Set(Concept)=Set{}|a->union(x.PREREQUISITE))
            -SEQUENCED_AFTER
                ->iterate(y:Topic; b:Set(Concept)=Set{}| b->union(y.concepts))
            -self.learningContext.ls->iterate(x:LearnerStereotype;
                a:Set(Concept)=self.learningContext.ls->first().constraints
                ->select(oclIsOfType(PresumedKnowledge))
                ->iterate(i;res:Set(Concept)=Set{}|res->including(i.competency.c)) |
                    x.constraints->select(oclIsOfType(PresumedKnowledge))
                ->iterate(j;res2:Set(Concept)=Set{}|res2->including(j.competency.c))
                ->intersection(a)
        = Set{}
        )
```

In the OCL constraint in Listing 6, we have defined an invariant which ensures that all goal concepts for all learners are covered somewhere in the courseware.

Listing 6. OCL invariant ensuring that the union of all learner stereotype goal concepts are covered in the courseware.

```
context CAVIAr
    inv all_learner_goals_covered: self.learningContext.ls->
            iterate(x:LearnerStereotype;
                a:Set(Concept)= Set{} | x.constraints->select(oclIsOfType(Goal))
                ->iterate(i:Goal; res:Set(Concept)=Set{}|
                    res->including(i.competency.c))
                ->union(a)
            - self.COURSEWARE_TOPICS->iterate(j; b:Set(Concept)=Set{}|
                b->union(j.TOPIC_CONCEPTS))
            = Set{}
```

This OCL rule constructs a set of all the goal concepts for all the learner stereotypes. The way the invariant does this is very similar to how the invariant in Listing 5 constructs a set of the common prerequisite concepts. In this case, the invariant iterates through each of the learner sterotypes and adds any goal concepts to a set, set G. This set is then returned and is compared with the set of all concepts covered in the courseware, set C. If all the goal concepts are covered by the courseware, the invariant is valid, i.e., the invariant is valid iff $G - C = \emptyset$.

4.2.3 Validating Courseware Adaptivity

As outlined in Section 3.4, CAVIAr provides for adaptivity by allowing the course creator to specify an entry constraint on topics in the courseware, where the entry constraint is defined in terms of learner knowledge and stereotype membership.

Courseware validation can be used to ensure that the courseware can adapt to a variety of different types of learners. For example, validation can check that each topic has supplementary material for learners who fail to achieve a set standard after delivery of the primary learning material.

An example case might be where the course creator wishes to check if there is additional learning material available for a learner who is struggling with a concept. In order to define this OCL constraint, we must firstly define what a "struggling learner" is, and how support is provided. For the purposes of this work, we define a "struggling learner" as a learner who has taken a courseware topic that covers some concept and is deemed to have a knowledge level of less than or equal to 0.3 in that concept. We define support for this learner as the provision of additional LOs, which cover the said concept and has a low or very low semantic density.

To ensure that this type of adaptivity is provided, the course creator needs to define a constraint to check for the existence of two topics, both covering the same concept and sequenced after one another, e.g., t_2 sequenced after t_1. t_1 has no entry requirements, while t_2 is only made available to learners who are struggling on the topic concept. This will ensure that there is material been made available for each concept, covered in the courseware, where a learner is struggling with that concept. We have defined this as an OCL constraint in Listing 7.

Listing 7. OCL invariant ensuring the existence of support material for learners struggling with a concept covered in a topic.

```
context Topic
    inv struggling_learner_supported:
        let sameConceptTopic : Topic = self.SEQUENCED_AFTER
            ->select(getTopicConcept()=self.getTopicConcept())->first()
        in
            self.sameConceptTopic.entryConstraint.lessThanCompetency.level <= 0.31
            and
            sameConceptTopic.resources
            ->select(oclIsOfType(LO)).educational.SemanticDensity < Scale::MEDIUM
```

4.3 Defining a Validation Model for an Instructional Design Theory

Instructional design theories "offer explicit guidance on how to better help people learn" [27] and can also be applied to ensure that a form of learning occurs, such as constructivism [28]. To illustrate how an instructional design theory is validated using CAVIAr, we outline the steps involved in validating that a given courseware uses Reigeluth's "Elaboration Theory" [29] correctly. Firstly, the elaboration theory is broken down into instructional principles, which must be true for the elaboration theory to be in use. The elaboration theory is defined as instructional principles in [29] as follows:

- Tasks are arranged from simple tasks to more complex tasks, starting with the simplest real-world version of the task moving to evermore complex versions of the task.
- Ensure tasks are not too big or too small.
- Ensure that any supporting content needed for tasks are available to the learner.
- Use the simplest version of tasks.
- Concepts are taught starting with the broadest concept and proceeds to ever more narrower and less inclusive concepts.
- Principles are taught starting with the broadest principle and proceeds to ever more narrower and less inclusive principles.

Once the instructional design has been formulated as instructional principles, we can then transform them into instructional constraints in the context of the CAVIAr model. We do this as follows:

- Division of CAVIAr courseware model into concepts, tasks and principles as follows:
  1. Each CAVIAr topic that delivers LO of type "Experiment" is a principle.
  2. Each LO, which is of type "Problem Statement" or "Exercise" is a task.
  3. All other CAVIAr topics are concepts, the concept is the most specialized concept that all LOs in the topic are related to.
- Sequence principles starting with the broadest principle according to the domain model, and progressively allow for the learner to see more specialized principles.
- Sequence tasks so that the simplest tasks, associated with the broadest domain concept, are sequenced first and then allow for them to get progressively more difficult.
- Sequence concepts according to the domain model, where the broadest concepts are sequenced first and then more specialized concepts.

When the course creator has specified the courseware constraints in terms of CAVIAr models, the constraints are then converted to OCL. The OCL validation model can then be validated by an OCL checker.

In Listing 8, we have defined the OCL constraints needed to validate the correct use of the elaboration theory in a courseware.

Listing 8. OCL defining instructional constraints ensuring correct application of elaboration theory.

```
context Topic
    def: isTask() : Boolean = resources->select(oclIsTypeOf(LO))
        ->exist(metadata.educational.LearningResourceType
            =LearningResourceTypes::PROBLEM_STATEMENT
        or e.metadata.educational.LearningResourceType
            =LearningResourceTypes::EXERCISE)

    -- All concepts covered after the current topic address narrower concepts
    inv topics_sequenced_according_to_ontology:
        GET_ALL_CONCEPTS_ASSOC_WITH_TOPIC->
        iterate(x:Concept; res:Set(Concept)=Set{}| res->union(x.NARROWER->asSet()))
        ->iterate(i:Concept; res2:Set(Topic)=Set{}| res2->union(i.CONCEPT_TOPICS))
        ->iterate(j:Topic; res3:Set(Topic)=Set{}| res3->union(j.SEQUENCED_AFTER))
            ->contains(self))

    -- easiest tasks no sequenced after constraint, all others have a
    -- sequenced after constraint
    inv task_sequencing:
        self.LOWEST_DIFFICULTY_CONTAINED_TOPIC.SEQUENCED_AFTER.oclIsUndefined() and
        self.CONTAINED_TOPICS->reject(self.LOWEST_DIFFICULTY_CONTAINED_TOPIC)
        ->forAll(not SEQUENCED_AFTER.oclIsUndefined())
```

5 Integration with the State of the Art in Adaptive Courseware Construction

As we have mentioned, courseware validation is a courseware construction activity. Our approach to adaptive courseware validation is neutral of a courseware construction methodology. To increase the value of adaptive courseware validation, it can be integrated into an existing courseware construction methodology. In this section, we will outline how our courseware validation approach can be integrated with the state of the art in adaptive courseware authoring tools. This is done by mapping the authoring tool's internal data models, representing the courseware construction concerns, to CAVIAr. In this section, we provide details on how this can be achieved through model transformation technologies.

The section begins by outlining what model transformations are and how they allow for interoperability between CAVIAr and the state of the art in courseware authoring tools. We then exemplify this by outlining how adaptive courseware developed using the MOT can be validated using CAVIAr.

5.1 Model Transformations

Model transformations allow for the transformation from one model type to another model type using a declarative transformation mapping defined between two metamodels. A model transformation specification defines how metamodel element(s) from one metamodel are mapped to metamodel element(s) of another metamodel. Transformation mappings are subjective and mapping alternatives could be defined by the course creator to replace the ones expressed here.

In order to define a transformation from an adaptive courseware specification to CAVIAr, firstly, a metamodel must exist for the adaptive courseware specification, secondly, a mapping must be defined from the adaptive courseware specification metamodel to the CAVIAr metamodel, and thirdly, any CAVIAr model elements not available through the adaptive courseware specification must be provided by the course creator.

Transformations separate the integration concern from the other concerns in courseware construction. This means that should a courseware specification change or a new one developed, the transformation just needs to be updated or defined, respectively.

The course creator uses the transformation to generate the CAVIAr models. Transformations are defined by model transformation experts to be used by the course creator to allow for interoperability from courseware specifications and knowledge representation standards to CAVIAr.

5.2 Case Study: Integrating of MOT with CAVIAr

To validate our approach to CAVIAr integration, we have tested it on AEH developed using the MOT tool, where AEH can be viewed as small-scale adaptive courseware delivered through hypermedia. We have chosen to validate the AEH produced by MOT, as it is portable and in a well-established AEH format, known as LAOS. Interoperability between AEH produced by MOT and CAVIAr is very challenging as there is a mismatch

between the specifications at the conceptual level. In [30], we provide extensive details on how these difficulties were overcome. In this paper, we will just outline the main points of the integration.

The MOT tool uses the LAOS model for defining AEH. Some preliminary work has been done in developing a metamodel for LAOS. Static elements of the LAOS model have been extracted for integration with AEH delivery systems in the form of an XML schema, known as the Common Abstraction Framework (CAF) [2]. We investigated what was involved in firstly mapping CAF to CAVIAr, and subsequently looked at mapping adaptive rules in LAOS to CAVIAr.

A formal CAF metamodel was required for the transformation definition. This was generated from the CAF XML schema. Tool support for this was provided by the Eclipse EMF framework [31].

Once the CAF metamodel was defined, the transformation between the CAF and CAVIAr metamodels was defined using a model transformation language [32]. Below, we outline the transformation mappings to each CAVIAr model.

In order to define the CAVIAr domain model, we define a mapping from the LAOS domain model concept map to the CAVIAr domain model. In this transformation, the CAF domain model *concept* is related to the CAVIAr domain model *concept*. The conceptual composition relationship in CAF, which relates two CAF concepts together, is transformed to the CAVIAr *ConceptRelationship* class of type *NARROWER*, where the contained concepts in LAOS are "narrower" in scope to that of the containing concept.

The CAVIAr learning context model is defined using the LAOS goal and constraint model definition. Mapping is defined as follows:

- The LAOS goal and constraints model is transformed into a single generic learner stereotype in CAVIAr.
- LAOS lesson goals are transformed into CAVIAr goals for the generic learner stereotype.
- Conceptual sequencing data in the LAOS lesson are transformed to *PRE_REQUISITE* relationships between concepts in CAVIAr.

An explicit courseware model is not defined in the CAF model but can be derived using the CAF's domain model. In LAOS, the domain model contains learning content in concept attributes. We define a transformation mapping for each concept in the CAF domain model to courseware topics in the courseware model.

In defining the transformation from the CAF model to the CAVIAr courseware model, we specify a 1:1 relation between the concepts in CAF and the CAVIAr courseware topics. CAF Concepts that are contained in other concepts are transformed to CAVIAr topics which have a *TopicRelationship—PART-OF* to the CAVIAr topic which has been mapped from the CAF containing concept.

There is generally a direct mapping from the LO metadata in courseware to the CAVIAr learning resource model, due to the extensive use of the IEEE LOM standard in the definition of the CAVIAr learning resource model. In some AEH tools, LOs are not explicit units of instruction in the courseware but are embedded in the domain model attributes. This is the case in LAOS. The learning resource model is therefore derived from the domain model definition. To generate LOs from the LAOS, we transform each conceptual attribute to an LO. The LO metadata are automatically derived for each LO generated, using the attribute type (e.g., title, conclusion) and the concept the attribute is associated with.

In the CAVIAr courseware model, adaptive behavior is principally provided by specifying restrictions on the sequencing of topics and/or restrictions on learner profiles that can access a topic. In LAOS, adaptive rules are defined using an adaptive rule language known as LAG [12]. In [30], we outline how the LAG adaptive rule abstract syntax is used to create a MOF metamodel for adaptive rules. A transformation mapping is then defined from the adaptive rule metamodel to the CAVIAr courseware metamodel.

Once the LAG and CAF have been transformed into CAVIAr C4 models, a validation model can be defined for the AEH courseware as outlined in Section 4.

6 Evaluation

In this section, we firstly evaluate the CAVIAr metamodel assessing its soundness and completeness, and then assess its expressiveness. After this, we evaluate our approach to adaptive courseware validation—constraint-based validation. The feasibility of the approach is validated by outlining details of a CAVIAr implementation, a courseware validation software tool. The subsequent sections outline how this software tool was used to evaluate user acceptance, performance, and usability of constraints-based courseware validation.

6.1 CAVIAr Soundness and Completeness

In order to evaluate the CAVIAr metamodel, we wish to establish its soundness and completeness. Guizzardi et al. outline how this can be done by comparing the metamodel to a well-established domain conceptualization such as an ontology [33]. Guizzardi et al. define a modeling language as *sound* when every modeling primitive has a representation in the ontology. They define a language as *complete* when every concept in the ontology is represented in the modeling language.

To establish the soundness and completeness of the CAVIAr metamodels, we mapped each of the metamodels, defined in CAVIAr, to a well-defined conceptualization representing the metamodel's course construction concern. In Section 5, we have outlined the degree of soundness and completeness of the CAVIAr domain model and learning context model by mapping them to the LAOS model, a well-established AEH specification for defining AEH requirements.

As we have already noted, the LAOS model does not represent courseware or LOs as found in CAVIAr. The courseware model can be mapped to either the ADL SCORM 2004 [25] specification or the IMS LD [18] specification. In our initial experiments, we have found that all elements in the courseware model can be mapped to some modeling element or group of modeling elements in both specifications. We can then conclude the CAVIAr courseware

metamodel to be sound. The completeness of the CAVIAr courseware metamodel is undetermined. Further experiment is required to establish if all combination of constructs in the LD and SCORM specifications can be represented in the CAVIAr courseware metamodel definition.

The IEEE LOM standard [24] was used as a domain conceptualization for the LOs in the CAVIAr learning resource model. The CAVIAr learning resource model is based on the IEEE LOM standard. There is therefore a one-to-one mapping of the elements in IEEE LOM to the modeling elements defined in the CAVIAr metamodel. We can therefore conclude that the learning resource model is both sound and complete.

6.2 CAVIAr Expressiveness

In evaluating the constraint-based approach outlined in this paper, we have carried out various case-study evaluations, where instructional design theories and models have been taken from [34] and defined in terms of a CAVIAr validation model. Tests have demonstrated that CAVIAr is expressive enough to capture the essence of instructional design theories as well as core personalization strategies. Sections 4.3 and 5 illustrate examples of such tests. By expressing well-established instructional design theories as CAVIAr constraints, we have shown our approach to be capable of checking courseware for real-world problems.

6.3 Feasibility

To validate our approach, we have designed and implemented a courseware validation tool, based on the Eclipse Modeling Framework (EMF) [31] that allows for OCL-based model validation and also model transformation through the Atlas Transformation Language (ATL) [32]. CAVIAr models are defined using EMF. An intuitive Graphical Modeling Framework (GMF) user interface has been developed for the course creator to view and edit the CAVIAr models. Our tool allows for the importing of SCORM 2004 and IMS LD courseware creating a CAVIAr courseware model. Domain models can also be imported from knowledge organization structures on the Semantic Web such as SKOS [21]. The tool provides the course creator with an intuitive-model-based interface with which to inspect and edit the CAVIAr models generated from import (transformation).

6.4 User Trials

In order to gauge the usefulness of CAVIAr validation, we have demonstrated courseware validation, using the software tool described in Section 6.3, to the course creator. The demonstrations were conducted in a one-to-one tutorial-type scenario. The CAVIAr validation model used had an constraint for each of the types defined in Section 4. There were 14 participants of the trial, all of whom have been responsible for authoring or adapting accredited courses in academia and/or industry. All considered themselves to be familar or an expert with e-learning, while 71 percent considered themselves familiar or expert in e-learning authoring, 86 percent were familiar with personalized e-learning, no participant considered themselves an expert.

The reaction to validation was very positive from all participants, with 92 percent of participants believing that validation exposed actual problems in the courseware. All participants agree, with 57 percent strongly agreeing, that they would be more confident in newly created courseware once it had been validated.

Many participants saw key advantages to the model-based validation, as validation constraints could be used to ensure that a given courseware meets some specified requirement, either for accreditation purposes or for legislative requirements.

Model transformation was seen as a flexible technology allowing for integration with the state of the art. Course creators did not always agree with the mappings defined but were satisfied with the option of altering the mapping in the transformation definition.

6.5 Performance

CAVIAr courseware validation uses an OCL checker that iterates through each model element instance checking if imposed constraints are violated. The processing time of CAVIAr is therefore dependent on the number of model elements in the CAVIAr model. This can be compared with other validation approaches where a learner's progression though courseware is simulated. These approaches are outlined in Section 7. The processing time of these approaches depend on the number of independent paths in courseware, which can be very large for personalized courseware. When conducting our user trials, we found that 57 percent of participants did not feel they were waiting a long time for their validation results, while the remainder gave a neutral answer.

6.6 Ease of Use

In the user trials, the definition of an OCL validation model was identified as a key usability concern. To this effect, we have raised the level of abstraction at which the course creator creates a validation model by developing software support for creating the validation model. This allows the course creators to create an OCL validation model using an intuitive domain specific modeling language [35]. Further to this, we anticipate constraint models to be reused in a similar fashion to LOs, limiting the need for the course creator to define their own validation model.

7 Related Work

The use of logics for validating courseware has been investigated by the ALICE project at the University of Torino [9]. The project looks at a range of course construction activities including course verification [36] and construction [9].

Courses are represented using action theory, where each course component is an action with preconditions and postconditions. Traditional AI reasoning, such as temporal projection, are used to check that all preconditions in the action theory are respected.

Curricula models allow for restrictions and constraints to be placed on possible learning resource sequences. Curricula models are formalized using temporal constraints and are independent of the learning resources, operating at the knowledge level. For example, a possible constraint might

be that $knowledgeElement_a$ must be learned before knowledge element $knowledgeElement_b$ can be attempted. Using linear-time temporal logic (LTL) to represent temporal constraints allows for the verification of the curriculum.

The motivation behind Baldoni et al.'s work is to validate Italian student study plans. A study plan is a list of courses a student takes in University. Each year students may alter their study plan. These alterations may have adverse effects to the students overall learning goal.

Verifying compliance means that the curriculum respects the constraints at the knowledge level represented using the curricula model, constraints at the resource level represented using action theory, and that the curriculum allows the learner to reach some goal state.

In order for logics to be a viable method for curriculum construction and verification, the logic coding would have to be hidden from the user. The work by Baldoni et al. looks primarily at the sequencing of modules in a degree style programme. Our work concentrates on the sequencing of topics within a degree module, the granularity level is smaller. Our work also looks to not only validate the sequencing of topics in courseware, but looks at other instructional concerns, as outlined in Section 4.

The Concept-based Courseware Analysis tool (CoCoA), developed at Carnegie Technology Education, uses two types of validation: typed items and advanced concept roles [8].

Typed items allow for validation of the positioning of particular teaching operations.

Advanced concept roles define a LO with regard to prerequsite knowledge and knowledge outcome. A concept has two types of prerequisites and two types of outcomes, strong and weak. A strong prerequisite or outcome indicates that deep knowledge of the concept referenced is required for the specified learning item, while a weak prerequisite or outcome indicates that only surface knowledge of the concept referenced is required.

CoCoA checks only sequential learning paths through learning material. This is done by simulating a learner's progression through the learning material. The tool then generates a report once all simulations are complete This report will indicate any "content holes," where the learner encounters an LO without having the necessary prerequsite knowledge needed to use the LO.

CoCoA validation goes beyond just validating basic course sequencing, the tool also has a variety of rules relating the type and location of learning items to each concept in a course.

CoCoA was prototypical in nature and as such there is no consideration for TEL standards. Pedagogical problems are defined by the CoCoA developer, there is no facility for the course creator to manipulate the validation rules. This caused problems with user acceptance [8]. CoCoA was not developed using an extensible architecture which would allow for the inclusion of unforeseen pedagogical rules in the future. It also does not reflect the complexity of validating the modern course as all courses validated using CoCoA must be linear in nature with no branching points.

8 Conclusions

In this paper, we have defined CAVIAr, a formal modeling framework used to express courseware in terms of its design and its requirements. We illustrated how valid courseware can be defined in terms of CAVIAr constraints using OCL. Constraint-based validation of courseware is a novel approach to courseware validation, where constraints are defined on courseware modeling constructs. This differs to the state of the art in validation, covered in Section 7, where validation is based on a simulation of learner progression through every possible learning path in a given courseware. This can lead to computational problems where there are many learning paths through courseware, such as that found in personalized courseware. Our approach is better suited to personalized courseware as it is not adversely affected by many learning paths. This is because our approach validates courseware in terms of its compositional structure rather than the possible learning paths it represents.

Our evaluation of constraint-based validation shows that course creators are more confident that adaptive courseware has been constructed correctly after validating it using CAVIAr. Validation reduces the risk of delivering pedagogically unsound courseware to a learner due to errors in personalization definitions.

We have also described how our approach to validating personalized courseware is completely interoperable with the state of the art in TEL and AEH specifications. This is achieved using model transformation technology. Model transformation technology is very flexible and can easily be extended to incorporate new specifications or changes to the existing ones.

At present, the CAVIAr constraints are represented in OCL, a language designed for software engineers, not course creators. We have outlined our work, creating a simple model-based user interface for creating validation constraints. We are also investigating the reuse of annotated instructional constraints. This way, the course creator does not, in general, have to define constraints but just looks for them in a constraint repository. A further extension to this would allow the grouping of instructional constraints into instructional designs, allowing the course creator to work at an even higher level of abstraction.

Validation, as presented in this paper, has been designed for university-level courses of a technical orientation. This type of course typically consists of lectures, tutorials, labs/practicals, and supplementary learning material. We would like to generalize our validation approach to consider other e-learning opportunities, such as (semi-)automatically generated courseware for self-learning. We would also like to consider the dimension m-learning would present to CAVIAr, such as considering the device the learner uses at delivery time.

References

[1] D. Dagger, "Personalised eLearning Development Environments," PhD dissertation, Univ. of Dublin, 2006.

[2] A. Cristea, D. Smits, and P. de Bra, "Towards a Generic Adaptive Hypermedia Platform: A Conversion Case Study," *J. Digital Information*, vol. 8, no. 3, http://journals.tdl.org/jodi/article/view/231/184, 2007.

[3] J.W. Samples, "The Pedagogy of Technology—Our Next Frontier?" *Connexions,* vol. 14, no. 2, pp. 4-5, 2002.
[4] P.V. Rosmalen, H. Vogten, R.V. Es, H. Passier, P. Poelmans, and R. Koper, "Authoring a Full Life Cycle Model in Standards-Based, Adaptive e-Learning," *J. Educational Technology and Soc.,* vol. 9, no. 1, pp. 72-83, 2006.
[5] J. Jovanović, "Generating Context-Related Feedback for Teachers," *Int'l J. Technology Enhanced Learning,* vol. 1, nos. 1-2, pp. 47-69, 2008.
[6] W. Dick and L.M. Carey, "Formative Evaluation," *Instructional Design: Principles and Applications,* second ed., pp. 227-267, Educational Technology Publications, 1991.
[7] L.M. Carey and W. Dick, "Summative Evaluation," *Instructional Design: Principles and Applications,* second ed., pp. 269-311, Educational Technology Publications, 1991.
[8] P. Brusilovsky and J. Vassileva, "Course Sequencing Techniques for Large-Scale Web-Based Education," *Int'l J. Continuing Eng. Education and Lifelong Learning,* vol. 13, nos. 1-2, pp. 75-94, 2003.
[9] M. Baldoni, C. Baroglio, and V. Patti, "Web-Based Adaptive Tutoring: An Approach Based on Logic Agents and Reasoning about Actions," *Artificial Intelligence Rev.,* vol. 22, pp. 3-39, 2004.
[10] D. Djurić, D. Gaševic, and V. Devedžic, "The Tao of Modeling Spaces," *J. Object Technology,* vol. 6, 2006.
[11] J.B Warmer and A. Kleppe, *The Object Constraint Language,* second ed. Addison Wesley, 2003.
[12] A.I. Cristea and M. Verschoor, "The LAG Grammar for Authoring the Adaptive Web," *Proc. Int'l Conf. Information Technology: Coding and Computing (ITCC '04),* vol. 1, pp. 382-386, Apr. 2004.
[13] A. Berlanga and F.J. García, "Towards Reusable Adaptive Rules," *Proc. Int'l Workshop Adaptive Hypermedia and Collaborative Web-Based Systems (AHCW '04),* July 2004.
[14] A.I Cristea, D. Smits, and P. De Bra, "Writing MOT, Reading AHA!—Converting Between an Authoring and a Delivery System for Adaptive Educational Hypermedia," *Proc. Third Int'l Workshop Authoring of Adaptive and Adaptable Educational Hypermedia,* http://wwwis.win.tue.nl/acristea/HTML/AIED05/5-10-cristea4-.pdf, June 2003.
[15] A.I Cristea and A. de Mooij, "LAOS: Layered WWW AHS Authoring Model and Their Corresponding Algebraic Operators," *Proc. 12th Int'l World Wide Web Conf. (WWW '03),* May 2003.
[16] P. De Bra and L. Calvi, "AHA!: A Generic Adaptive Hypermedia System," *Proc. Second Workshop Adaptive Hypertext and Hypermedia,* http://wwwis.win.tue.nl/ah98/Proceedings.html, Eindhoven University of Technology, June 1998.
[17] A. Moore, T.J. Brailsford, and C.D. Stewart, "Personally Tailored Teaching in Whurle Using Conditional Transclusion," *Proc. 12th ACM Conf. Hypertext and Hypermedia (HYPERTEXT '01),* pp. 163-164, 2001.
[18] H. Hummel, J. Manderveld, C. Tattersall, and R. Koper, "Educational Modelling Language and Learning Design: New Opportunities for Instructional Reusability and Personalised Learning," *Int'l J. Learning Technology,* vol. 1, no. 1, pp. 110-126, 2004.
[19] D. Hernández-Leo, E.D. Villasclaras-Fernández, J.I. Asensio-Pérez, Y. Dimitriadis, I.M. Jorrín-Abellán, I. Ruiz-Requies, and B. Rubia-Avi, "Collage: A Collaborative Learning Design Editor Based on Patterns," *Educational Technology and Soc.,* vol. 9, no. 1, pp. 58-71, 2006.
[20] M. Melia and C. Pahl, "An Information Architecture for Validating Courseware," *Proc. First Int'l Workshop Learning Object Discovery and Exchange (LODE '07),* Sept. 2007.
[21] A. Miles and D. Brickley, "SKOS Core Guide," Technical Report, 2005.
[22] R. Gagné, W. Wager, K. Golas, and J. Keller, *Principles of Instructional Design,* fifth ed. Wadsworth, 2005.
[23] N. Stash A. Cristea, and P. DeBra, "Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions," *Proc. 13th Int'l World Wide Web Conf. Alternate Track Papers and Posters (WWW Alt. '04),* pp. 114-123, 2004.
[24] "LTSC WG12: Learning Object Metadata," IEEE Learning Technology Standards Committee, 2002.
[25] A. ADL, "SCORM 2004 Overview," http://www.adlnet.gov/scorm/index.cfm, 2004.
[26] M. Melia, "Constraint-Based Validation of Courseware Using Model-Driven Engineering," PhD dissertation, Dublin City Univ., 2009.
[27] C.M. Reigeluth, "What is Instructional-Design Theory and How Is It Changing?" *Instructional-Design: Theories and Models, A New Paradigm of Instructional Theory,* vol. 2, pp. 5-30, Lawrence Erlbaum Assoc., 1999.
[28] R.E Mayer, *Designing Instruction for Constructivist Learning.* Lawrence Erlbaum Assoc., 1999.
[29] C.M. Reigeluth, ed., "The Elaboration Theory: Guidance for Scope and Sequencing Decisions," *Instructional Design: Theories and Models,* vol. 2, pp. 425-453. Lawrence Erlbaum Assoc., 1999.
[30] M. Melia and C. Pahl, "Towards the Validation of Adaptive Educational Hypermedia Using CAVIAr," *Proc. Sixth Int'l Workshop Authoring Adaptive and Adaptable Hypermedia (A3H '08),* July 2008.
[31] F. Budinsky, S.A. Brodsky, and E. Merks, *Eclipse Modeling Framework.* Pearson Education, 2003.
[32] F. Jouault and I. Kurtev, "Transforming Models with ATL," *Proc. Model Transformations in Practice Workshop (MoDELS '05),* Oct. 2005.
[33] G. Guizzardi, L.F. Pires, and M. van Sinderen, "Ontology-Based Evaluation and Design of Domain-Specific Visual Modeling Languages," *Proc. 14th Int'l Conf. Information Systems Development,* pp. 691-705, 2005.
[34] C.M. Reigeluth, *Instructional Design: Theories and Models,* vol. 2. Lawrence Erlbaum Assoc., 1999.
[35] U.T Janjua, "Model Based OCL Generation," MS thesis, Dublin City Univ., 2008.
[36] M. Baldoni, C. Baroglio, V. Patti, and L. Torasso, "Reasoning About Learning Object Metadata for Adapting SCORM Courseware," *Proc. Int'l Workshop Eng. Adaptive Web: Methods and Technologies for Personalization and Adaptation in the Semantic Web (EAW2 '04),* pp. 4-13, Aug. 2004.

Mark Melia received the BSc degree in software systems from the National College of Ireland. He is currently working toward the PhD degree at Dublin City University within the Software and Systems Engineering Research Group. His research interests are in the area of courseware construction and courseware modeling. He has published numerous papers on this topic. He is a student member of the IEEE.

Claus Pahl is a senior lecturer and the leader of the Software and Systems Engineering Research Group at Dublin City University, which focuses on Web technologies and e-learning applications in particular. Dr. Pahl has published more than 160 papers, including a wide range of journal articles, book chapters, and conference contributions on e-learning. He is on the editorial board of the *International Journal on E-Learning* and the *International Journal of Technology-Enhanced Learning*, and is a regular reviewer for journals and conferences in the area of software, Web, and learning technologies and their applications. He has extensive experience in educational technologies, both as an instructor using technology-supported teaching and learning at the undergraduate and postgraduate level and as a researcher in Web-based learning technology. The IDLE environment, developed by him and his students, has been in use in undergraduate teaching since 1999. He is a member of the IEEE.