An Information Architecture for Validating Courseware

Mark Melia and Claus Pahl

School of Computing, Dublin City University, Dublin 9, Ireland
mmelia@computing.dcu.ie

Abstract. Courseware validation should locate Learning Objects inconsistent with the courseware instructional design being used. In order for validation to take place it is necessary to identify the implicit and explicit information needed for validation. In this paper, we identify this information and formally define an information architecture to model courseware validation information explicitly. This promotes tool-support for courseware validation and its interoperability with the courseware specifications.

1 Introduction

The assembly of Learning Objects (LOs) into courseware is increasingly becoming the norm in courseware construction. LO reuse offers course creators an efficient methodology to create courseware from tried and tested components. This methodology also has challenges. One of the most pressing challenges the course creator faces is placing the right LO in the right place in courseware [8]. This is difficult as course creators typically only have high level knowledge of the content and internal behaviour of each LO, due to 3rd party construction and increasing internal complexity.

Quality courseware requires a consistent and suitable instructional design. Due to a lack of understanding, a course creator may place LOs in courseware that are not compliant with courseware's instructional design. This causes inconsistent pedagogy which can confuse, demotivate and isolate the learner and could ultimately lead to the rejection of the course by the learner [7].

In this paper, we detail the courseware information that must be captured in order to accurately validate courseware developed using LOs and define a layered information architecture to that effect. The formal definition of the information architecture allows for a validation engine to be built around it, providing tool support to the course creator when validating courseware. Our formal definition also facilitates for translation to and from external courseware representation specifications such as SCORM [1] and IMS LD [5], allowing for interoperability.

2 Identification of Information Needs

Courseware validation is a complex task which involves evaluating courseware in the context of its scope (the knowledge the courseware wishes to teach), instruc-

tional approach (how it teaches this knowledge) and content used (educational materials used in teaching). In the real world this may mean delegating validation tasks to experts. Validation of course structure may be delegated to an instructional design expert, while content related issues, may be delegated to a subject matter expert.

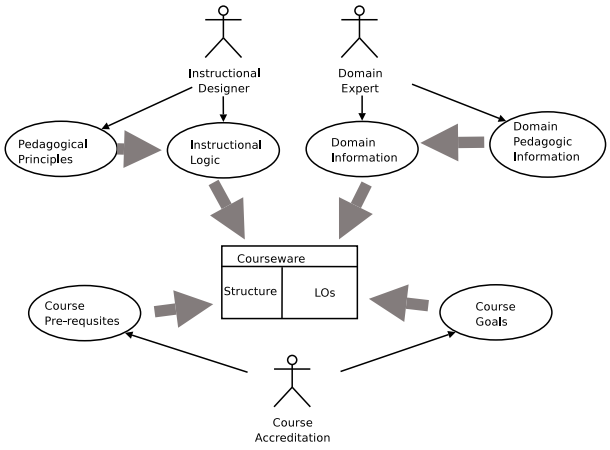

Fig. 1. Courseware Validation Concerns

In order for the courseware validation process to be automated, the implicit knowledge and information used to validate courseware, known as the "courseware validation concerns", must be identified and defined. Courseware information changes over time (e.g. a more detailed learner model is possible post-delivery) definitions must focus on one stage of the courseware life-cycle. Here we define courseware information at the post-construction/pre-delivery stage of the courseware life-cycle (i.e. after the courseware has been developed by the course creator but before it is delivered to learners).

Fig. 1 outlines the courseware validation concerns for the post-construction/pre-delivery stage of the coureware life-cycle. Three expert roles are represented in the figure, the domain expert, the instructional designer and course accreditation. These roles may be embodied by the one course creator, or may represent several specialised course creators. Domain information outlines the knowledge to be taught to the learner. The domain expert can also define basic rules of thumb on how particular domain knowledge is taught (e.g. concept A always before B). The scope of the courseware is defined in the context of the domain information specifying the course pre-requisites and the course goals. The course scope is usually defined by the course accreditation body. The instructional design defines a strategy for the transfer of knowledge to the learner and is derived from general instructional design principles.

Courseware is defined as instructional logic combined with LOs. LOs are the content which teach aspects of domain knowledge to the learner and are described using metadata, outlining the material covered by the LO and the method of instruction. Instructional logic specifies how the learner proceeds through the courseware, from one LO to another.

3 Layered Information Architecture

3.1 Information Architecture Overview

By explicitly representing the courseware validation concerns identified in the previous section it is possible to automatically validate courseware constructed. The Courseware Authoring Validation Information Architecture (CAVIAr) allows for the representation of the validation concerns using a system of layered models. CAVIAr extends the LAOS model, used in Adaptive Educational Hypermedia (AEH) authoring. LAOS simplifies AEH authoring by separating AEH concerns allowing the course creator to deal with each individually. [3]. Our model replicates the benefits of using the LAOS model in AEH authoring by separating the courseware validation concerns.

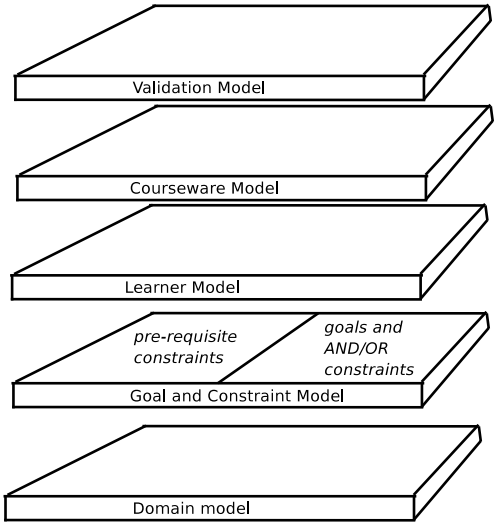

Fig. 2. The Courseware Authoring Validation Information Architecture (CAVIAr)

The bottom three layers of the validation model, the domain model, the goal and constraint model and the learner model, are replicated from the LAOS model but adapted for validation. On top of these base layers the courseware model and validation model are defined.

3.2 The Domain Model

The domain model is a formalism of knowledge in the form of a concept map, where a node represents a concept and an edge represents the relationship from one concept to another.

A conceptual graph for domain modeling in AEH authoring is formally defined in [3]. In AEH the domain model serves as the main navigational tool, while in courseware validation it serves as a semantic point of reference for LO annotation (section 3.5).

We can define the courseware validation domain model requirements as follows:

Definition 1. *A concept map CM is determined by the tuple* $<C, L>$ *where* C *is a set of concepts and* L *is a set of links.*

- *A concept* $c \in C$ *is an abstract notion which is described using attributes. There must be at least one attribute for each concept, the name of the concept described as* n_c.
- *A link* $l \in L$ *is a tuple* $<c1, c2, t_l, n_l>$ *with* $c1 \in C$, $c2 \in C$ *start and end concept respectively,* n_l *is the name of the link. There are many types of links, the link type adds semantics to the link, indicating what is meant by the link.* t_l *refers to the link type (e.g. "is-a", "has-a").*
- *Each concept c must have at least one link, the hierarchical link, which links a concept to its parent concept. An exception of this is the root concept.*

3.3 The Goal and Constraints Model

The purpose of the goal and constraints model in CAVIAr is to specify domain pedagogic information and course conceptual goals. The course creator can express which concepts from the domain model are to be taught to the learner and conceptual pre-requisite constraints between concepts.

We distinguish between two types of knowledge, deep knowledge and shallow knowledge, where deep knowledge implies an understanding of a concepts underlying concepts or sub-concepts and shallow knowledge implies a passing knowledge or a mere familiarity with a concept. To this effect, when defining a model in terms of knowledge (i.e. the domain model), we must distinguish if we mean deep knowledge or shallow knowledge. To do this, goals and constraints have a strength - weak or strong, which specify deep knowledge or strong knowledge respectively. This accommodates the needs of real courses as specified in [2].

We now give a more formal definition of the elements which make up the goal and constraint model.

Definition 2. *The goal and constraint model is made up of the tuple* $<P, G>$, *where* P *is a set of pre-requisite constraints, which specify a relationship between concepts where one concept must be understood before the other concept and* G *is a set of Goals.*

- *A pre-requisite constraint $p \in P$ is described using the tuple $<c1, c2, ps>$, where $c1 \in C$, $c2 \in C$ and ps describes the strength of the pre-requisite constraint, weak or strong. Weak pre-requisites require only passing knowledge of the pre-requisite concept, a strong pre-requisite requires deep knowledge of the pre-requisite concept and its underlying concepts.*
- *A goal $g \in G$ is described by the tuple $<GC, alt, g_g>$, where GC is a set of goal concepts, alt defines alternative goal and g_g is a nested goal within the goal g.*
- *A goal concept $gc \in GC$ is made up of the tuple $<c, gs>$, where c refers to a concept in the domain model $gc \in C$ and gs is the strength of the goal, a weak goal or a strong goal.*

We can infer that the goal and constraint model is an overlay model on the domain model in that the goal and constraint model is specified in terms of the domain model. This is due to each $gc \in C$, meaning that $GC \subseteq C$ and pre-requisite constraints are expressed using concepts from the domain model.

3.4 The Learner Model

In CAVIAr, the learner model is used to represent the stereotypical learner knowledge or pre-requisite knowledge for a courseware. The learner model is a necessary layer of CAVIAr as concepts not covered in the courseware may be pre-requisite of concepts in the courseware, but this knowledge is assumed learner knowledge. In order for a validation engine to acknowledge the learner's initial assumed knowledge it must be modelled.

As we specify the learner model in terms of conceptual knowledge the learner model elements also have a strength, weak or strong, as is the case for goals and constraints.

We can also describe the learner model formally as follows:

Definition 3. *The learner model is described by the tuple $<kc, ks>$, where kc refers to a concept in the domain model $kc \in C$ and ks is the strength of the assumed knowledge, weak or strong.*

The learner model is also an overlay model on the domain model as each $kc \in C$, again meaning $kc \subseteq C$.

3.5 The Courseware Model

A course structure can be modeled as a directed graph. Learning resources are nodes on the graph. A learner's traversal through the graph's edges depends on variables, such as learner choice, assessment results, learning styles, feedback, which are assessed at run-time. Learning resources in the course are annotated using some metadata standard such as IEEE LOM [6]. The metadata maps the learning resource to the concept(s) it addresses in the domain model.

Fig. 3 demonstrates how each LO in a course is associated with at least one concept in the domain model, specified in the LO annotation (LO conceptual

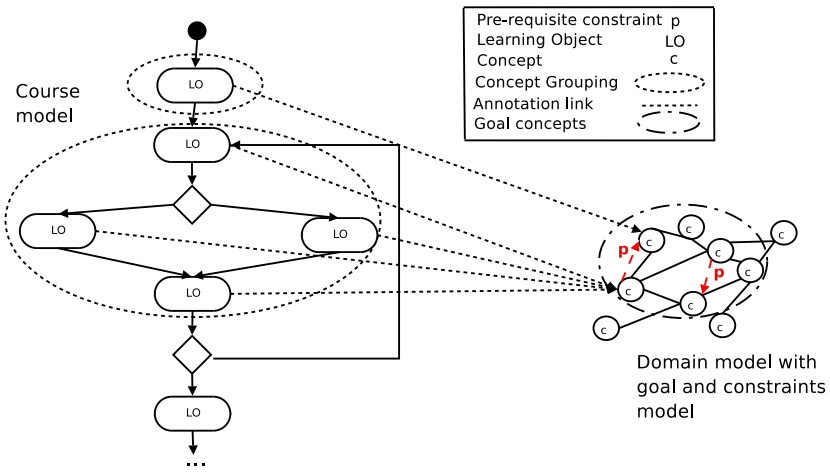

Fig. 3. Grouping LOs according to the concept they cover

annotation is indicated in the diagram with an arrow from a LO to a concept). LO conceptual associations can be used to group LOs. Grouping are made up of LOs concerned with the same concept. This type of LO grouping is illustrated in the diagram in figure 3 (dotted circle on course model indicates conceptual groupings). Once we can group LOs by concept we can discriminate between pedagogical strategies concerned with inter-conceptual pedagogy (strategy concerning sequencing of the conceptual groupings) and intra-conceptual pedagogy (pedagogical strategy concerning the LOs within each conceptual grouping).

We now give a more formal definition of the course model.

Definition 4. *We consider courseware CW to be determined by the tuple* $<LO, LP, SP, EP>$*, where LO represents a set of Learning Objects and LP is the set of learning paths. SP and EP represent the start points and end points respectively in a given courseware model.*

- *A learning path* $lp \in LP$ *is defined by the tuple* $<lo1, lo2, G>$*, where* $lo1 \in LO$*,* $lo2 \in LO$ *start and end learning objects respectively. G refers to a boolean gate condition, which must be true for the learner to proceed down that learning path.*
- *A learning object* $lo \in LO$ *is defined by the tuple* $<LC, A_{lo}>$*, where LC is learning content and* A_{lo} *is the LO's annotation.*
- *A learning object's annotation* A_{lo} *is a tuple* $<M, CA>$*, M is a metadata set used to describe a LO. CA is a set of conceptual annotation.*
- *Metadata m where* $m \in M$ *is a tuple* $<att, val>$*, where att is the name of the metadata attribute and val refers to its value.*
- *A Conceptual Annotation* $ca \in CA$ *is a tuple* $<pur, c, s>$*, where pur is the purpose of the annotation (i.e. to specify a competency in a particular con-*

cept), c∈C a reference to domain concept and s the strength of the link to the concept.

The connection between the courseware model and the lower layers is in the LO's annotation. Each LO's annotation points to a concept c, where $c \in C$.

In our definition we have defined a LP with a boolean gate condition, G. This gate condition allows the course creator to specify what learning path the learner should take based on various variables found in e.g. the learner model.

3.6 The Validation Model

The Validation Model captures pedagogical rules that courseware must adhere to. In the validation model layer the course creator can express undesirable properties of courseware and also properties which must be present in the courseware.

Validation can be split into two distinct parts, validation which is concerned with learning content of one domain concept (i.e. intra-conceptual validation) and validation which looks at how the course proceeds from one concept to another in the course (i.e. inter-conceptual validation). Typical intra-conceptual validation will ensure that each concept is taught in a uniformed manner, while inter-conceptual sequencing might ensure that the strategy undertaken in the course is one of depth-first (covering each concept in detail before moving on to the next concept).

The formal definition of the lower four layers of CAVIAr allows for the formation of pedagogical rules. Here we specify a rule which states for every concept c that is a goal concept, there must exist a LO annotated with that concept which is of "Type" "Lecture".

$\forall x \exists y(c(x), c \in gc_i) \wedge (CW_i.LO(y).A.CA.c = x) \wedge (CW_i.LO(y).A.M(z).Att ='' Type'') \wedge (CW_i.LO(y).A.M(z).value ='' Lecture'')$

In the example below, we demonstrate how the rule above can be implemented using the JESS rule language, a Java-based, Lisp-like, rule language [4]. The rule states that each concept in the goal model, "concepts-in-gm", must have a LO which is annotated to have a "resource_type" of "lecture". If this is not the case the rule prints out the violating concept with the message "HAS NO LECTURE".

```
(defquery findConceptsLOs    //QUERY: find the LOs associated with the given concept
  (declare (variables ?c))
  (concept-to-lo (concept ?c) (lo ?lo)))

(defquery findLoResourceType  //QUERY: Get the LOs with the given resource type
  (declare (variables ?x ?resourceType))
  (lom (id ?x) (resource_type ?resourceType)))


(foreach ?c ?concepts-in-gm   // for each concept in goal model
  (bind ?list (new java.util.ArrayList))
  (bind ?los-in-concept (new java.util.ArrayList))
  (bind ?concept-ok FALSE)
  (bind ?itr(run-query findConceptsLOs ?c)) //get LOs for concept
  (while(?itr hasNext)  //for each LO
    (bind ?token (call ?itr next))
    (bind ?fact (call ?token fact 1))
    (bind ?lo (fact-slot-value ?fact lo))
    (?list add ?lo) // add LO to list
  )
  (foreach ?l ?list  //for each LO in list
    (bind ?itr2(run-query findLoResourceType ?l lecture))  //does LO have resource
```

```
                                                                    type lecture
    (if (?itr2 hasNext) then
      (bind ?concept-ok TRUE) //if it does then concept ok
    )
  )
  (if (not ?concept-ok) then //if the ?concept-ok variable has not been changed - no
                                                                    lecture type
    (printout t "CONCEPT "?c" HAS NO LECTURE" crlf)
  )

)
```

4 Discussion

In this paper we have given an overview of the information needs for the validation of courseware. We have then taken the information needs identified and defined a layered information architecture (CAVIAr), which allows for the explicit representation of each information need. The definition of each layer allows for the formulation of pedagogical rules, which can be validated in the context of our information architecture.

The formal definition of the CAVIAr allows for tool support to be built around it, in our future work, we will design and implement a validation engine based on CAVIAr. Another advantage of CAVIAr's formal definition is that it allows for translation from external courseware representations to CAVIAr, allowing for interoperability with existing courseware specifications, thus embracing the state of the art in courseware construction.

References

1. Advanced Distributed Learning. SCORM 2004 Overview, 2004. Available from: http://www.adlnet.gov/scorm/index.cfm.
2. P. Brusilovsky and J. Vassileva. Course sequencing techniques for large-scale web-based education. *International Journal Continuing Engineering Education adn Lifelong Learning*, 13(1/2):75–94, 2003.
3. A. I. Cristea and A. de Mooij. LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. In *Proceedings of The Twelfth International World Wide Web Conference (WWW03), Alternate Track on Education*. ACM, May 20th - 24th 2003.
4. E. Friedman-Hill. *JESS in Action: Java Rule-Based Systems*. Manning Publications, Greenwich, Connecticut, 2003.
5. H. Hummel, J. Manderveld, C. Tattersall, and R. Koper. Educational modelling language and learning design: New oportunities for instructional reusability and personalised learning. *International Journal of Learning Technology*, 1(1):110–126, 2004.
6. IEEE Learning Technology Standards Committee. LTSC WG12:Learning Object Metadata, 2002.
7. J. W. Samples. The pedagogy of technology - our next frontier? *Connexions*, 14(2):4–5, 2002.
8. D. A. Wiley. *The Instructional use of Learning Objects*, chapter Connecting Learning Objects to Instructional Design Theory: A definition, a methaphor and a taxonomy. Association for Educational Communications and Technology, 2001.