EDUCATIONAL COMPONENT MODEL FOR ADAPTIVE WEB-BASED COURSES

Freddy Duitama¹, Bruno Defude¹, Amel Bouzeghoub¹, Claire Carpentier¹

¹Department of Computer Science,
Institut National des Télécommunications
9, rue Charles Fourier - 91011 Evry cedex France

ABSTRACT
The primary business problem that a Learning Content Management System faces is to create just enough content, just in time, meeting the needs of different types of learners. A possible answer is the concept of educational component. This paper addresses the formal definition of an educational component model. Its framework will be the Learning Management System. Besides, we define both canonical operators to facilitate component composition and automatic metadata definition, and high-level constructors to coalesce and copy components. The goal is to provide a component-based environment of composition and delivering of platform independent educational contents to support adaptive learning. Additionally, we provide strategies to classify educational chunks, constituting a common base of knowledge and facilitating sharing of the learning objects.

KEYWORDS
Educational component, reuse of learning objects, Web-based adaptive learning system, educational ontology.

1. INTRODUCTION
A Learning Content Management System (LCMS) enables content provider to register, assemble, manage and publish learning objects. The primary business problem that a LCMS faces is to create just enough content, just in time, to meet the needs of different types of learners. Rather than developing entire courses and adapting them to multiple audiences, instructional designers create reusable content chunks and make them available to course developers throughout the organization. This eliminates duplicate development efforts and allows the rapid assembly of customized content.

The premise that independent chunks of learning objects can be developed to support technology-based learning has been support of multiple educational experiences. As the first works in the field of LCMS, the principal goals were fixed: adaptation (to learners), flexibility (rather than composition fixed) and scalability (industrial production without proportional cost). A possible answer to meet these three goals is the concept of educational component. This concept was already largely studied and certain standards exist several years ago: LOM (IEEE, 2002), Dublin Core (Mason, J., and Sutton, S., 2000), ARIADNE (Ariadne Foundation, 2002.). All these works converge to the idea of providing a framework for the exchange of learning objects among systems and the need of using metadata to describe them. Later, these ones were extended by the Reference Model SCORM (Advanced Distributed Learning Initiative, 2001), which is a suite of technical standards that enable web-based learning systems to find, import, share, reuse, and export learning content in a standardized way. However, SCORM does not fully take into account the semantics of the components and defines weakly sequencing and navigation capabilities in composed learning objects.

There are several experiences aiming to reusability and adaptability of learning objects. Repositories for learning objects have been built: Physlets – Physics Applets – flexible java applets designed for science education (Physics Applets, 2001). Wisconsin Online Resource Center provides small and independent chunks of knowledge or interactions stored in a database that can be presented as units of instruction or information (Wisconsin on-line center, 2000). The Merlot portal contains metadata of several learning object
EDUCATIONAL COMPONENT MODEL FOR ADAPTIVE WEB-BASED COURSES

Repositories (Merlot, 2002). These efforts have powered the rise of web-based learning environments and its associated emphasis on standard-based interoperability. The approach RLO/RIO (Barritt, C., 2001) (Reusable Learning Object/Reusable Information Object) defines a strategy to design and develop training contents (RLO) by combining several RLOS. (Downes, S., 2003) proposes a distributed model of learning object repositories to create an open and accessible marketplace for them. The Multibook project (El Saddik, A. et al., 2001) examines the current LOM standard and proposes extensions in order to match the specific constraints of multimedia content.

This paper addresses the formal definition of an educational component model. A Component model allows interaction among components that share consistent assumptions about what each provides and each required of the other. In our case, its framework will be the Learning Management System; where each educational chunk can be indexed, used independently, coupled with others, and reused in different contexts. Our approach allows both a semantic description of learning objects and to build platform-independent educational chunk. Additionally, we define operators to facilitate component composition and automatic metadata definition. Finally, we provide strategies to classify educational chunks, constituting a common base of knowledge to facilitate sharing of the teaching resource. Our aims are both to provide a component-based environment of learning object composition and to support customized delivering of content (i.e. adaptive sequencing and navigation capabilities).

This article is structured as follows: In section 2, we present system architecture or framework used by our component model. In section 3, we discuss the educational component model, the component composition, and canonical and high level constructors. The authoring and delivering process is described in section 4. Section 5 concludes the paper and presents some open issues.

2 SYSTEM ARCHITECTURE

This section presents the general architecture of an adaptive web-based learning system. It will be the framework for our component model. The architecture is based on the reference model for the adaptive hypermedia AHAM (De Bra, P., et al, 1999). In AHAM adaptation is based on a Domain Model (DM), a user model (UM), and a Teaching Model (TM), which consists of pedagogical rules. However, our approach uses educational components instead of the notion of pages or fragments. Besides, each component is classified under one or several concepts taken from the DM. In the following sub-sections, we will develop the DM and present briefly UM and TM.

2.1 Domain Model

We define an ontology for each specific domain (e.g. Computer Science, Biology, Physics) representing all domain concepts (see Figure 1). It is defined by a hierarchical description of important concepts in the domain. Besides, additional relations between concepts may be defined using rhetorical relationships.

Definition. Let the domain model be a graph \( G = <N, E> \), where \( N \) are nodes representing domain concept from the domain model and \( E \) are labeled edges representing relationship between two domain concepts. There are two kinds of possible relationships.

2.1.1 Hierarchical relationship

We define a hierarchy \( T_i \) for each specific domain. This one is defined using the broader/narrower relationship; i.e. "Concept A is broader than concept B whenever the following holds: in any inclusive search for A all items dealing with B should be found. Conversely B is narrower than A" (Fischer, D., 1998). Every domain concept inside the hierarchy \( T_i \), except root concept, must have a relationship "is narrower than" with one and only one domain concept. Every domain concept inside the hierarchy \( T_i \) can have zero or many "is broader than" relationship. The root in each \( T_i \) hierarchy is the most general domain concept and the final-nodes are the most specific domain concepts, i.e. nodes having zero is-broader-than relationships. The most specific concepts are defined in such a way that they can serve as section or units of instruction in a course.
2.1.2 Rhetorical relationship

Two domain concepts have a rhetorical relationship if this one exists independently of how both concepts are developed. There is a set of predefined rhetorical relationships taken from (Mann W. and Thomson S., 1987) Antitheses, Contrast, Extend, Background, Restatement, etc; however, they can be extended.

The DM will allow us both to define efficient searching and reasoning mechanisms to support an adaptive learning environment and to classify components by contents. On the other hand, if teacher needs, at authoring-time, to define a logical sequence for the concepts in a course, these relationships provide information about global coherence among different concepts. Finally, at learning time, they provide helps to learner searching complementary or related information about concepts.

2.2 User Model

An adaptive learning system may adapt contents depending on learners’ background, goals, and preferences. Our approach considers the three aspects. The UM is an overlay model. For each learner, we maintain an evaluation of his/her knowledge level for each domain concept. Users provide some information (e.g. his/her graphical preferences) whereas other one is captured or modified dynamically by analyzing his/her behavior (e.g. his/her level relative with respect to the domain concepts).

The UM = \{<learner, role, domain-concept, knowledge-level>\}, where learner corresponds to user-id. Role describes topics known by learner with respect to a domain concept; we take from educational ontology (Ranwez, S., et al, 2000) possible pedagogical roles played by a learning object; role ∈ \{introduction, definition, description, summary, application, ...\}. Domain-concept is a node from domain-model. In order to define the knowledge-level, (De Bra, P., et al, 1999) describes several modalities of knowledge representation. Most web-based learning systems to date use a discrete knowledge representation (Brusilovsky, P. et al, 1998), (Weber, G. and Specht, M., 1997), (El Saddik, A. et al, 2001), etc. Our approach uses the following set of discrete values \{not-visited, visited, very low, low, medium, high, very high\}. If we have no information about user, stereotypes are used.

2.3 Teaching Model

There are two possible teaching strategies: goals-based and course-based. The former provides to learners the closest contents with respect to their objectives, background and preferences. On the other hand, if learners need to work a course completely, system displays the entire conceptual map with annotations and helps them to meet course goals. TM includes content adaptation, link adaptation, and updating the UM.

3. EDUCATIONAL COMPONENT MODEL

This section presents both a formal definition of an Educational Component (EC) and several operators that support construction of more complex components. The semantic of these operators is described to determine rules of coherence when various components are combined. We extend well-known standards like SCORM to describe several characteristics addressed to support reuse of dynamic learning objects (El Saddik, A. et al, 2001).

The model covers different component types. a) Static and dynamic components. The former include the traditional static web pages; the lasts cover applications made available via Internet, which probably must be
set at deployment time (web services, multimedia resource, etc.). b) Primitive and composed components, i.e., non-divisible chunks and components built from other ones by aggregation.

3.1 Educational component definition

An Educational component is a unit of composition with an interface providing information about requirements for its use, coupling, or replacement during the technology-based learning. This unit can be used independently or for composition by third parties. It can be described by the tuple \(<\text{Coid}, \text{Metadata}, \text{Composition}>\) with the following definition:

a) \(\text{Coid}\): Component unique identifier.

b) \(\text{Metadata}\): It covers Input, Output, Prerequisites, Acquisition Function, Contents and Other-Characteristics.

c) \(\text{Composition}\): A composed component is described by an acyclic directed graph.

A dynamic component has inputs and outputs with the following definition:

a) \(\text{Input} = \{ I_i \}, i = 0...n\). Each \(I_i\) is defined in the same way as \(I\).

To classify a component the model uses \(\text{Contents} = \{R_i\} \neq \emptyset\), where \(R_i\) is defined by the tuple \(<\text{Coid}, \{\text{role}\}, \text{domain-concept}>\). \{\text{role}\} describes topics developed by a component with respect to a domain concept, \(\text{role} \in \{\text{introduction, definition, description, application, evaluation, demonstration, summary, ... }\}\).

Furthermore, each component can demand prerequisites (i.e. background required for using it successfully at learning time). \(\text{Prerequisites} = \{R_p\}\), where \(R_p = <\text{Coid}, \text{role}, \text{domain-concept knowledge-level}>\). Item \(\text{role}\) is optional, it means that can be unknown. In this case, it only defines a knowledge level with respect to the domain concept. \(\text{knowledge-level} \in \{\text{very low, low, medium, high, very high}\}\).

The Acquisition Function \(f(A)\) records how well students learn concepts developed by component. At learning time, for each couple \((\text{domain-concept, role})\) from \(\text{Contents}\) of \(\text{Coid}_i\): \(f(A)\) adds \(<\text{learner, domain-concept, role, new-knowledge-level}>\) in UM or \(\text{FAIL}\). Component creators can define \(f(A)\) using \(\{O_k\} \subseteq \text{Output}\) or using L.O.M Metadata Typical-Learning-Time versus User-learning-Time, etc. Note that \(f(A)\) defines a mapping between knowledge representation used by this model and knowledge representation used by teacher inside component.

Finally, \(\text{Other characteristics} = \{M_i\} \) with \(i = 0...k\) and \(M_i = <\text{tag, value}>\). They are used to describe non-functional component properties. For instance, L.O.M Metadata used to describe author, title, etc.

**Example**: Let \(C_2\) be a component identified by \(\text{Coid}_2\). It has \(I_1\) as \(\text{Input}\), \(\text{CT}\) as \(\text{Contents}\) and \(P\) as \(\text{Prerequisites}\). Let suppose \(I_1 = <\text{SetLanguage, Language, Possible values (English, French, Spanish)}>\); at deployment time, it will allow setting the language used by component; e.g., a video displays on Spanish among three possible languages. \(\text{CT} = <\text{Coid}_2, \{\text{introduction, definition, application}\} \text{Data-Model} >\) describes contents for component \(C_2\). It introduces, defines and applies the domain-concept \(\text{Data Model}\). Whereas, \(P = <\text{Coid}_2, \text{definition, First-Order-Logic, High}>\) states that component \(C_2\) requires preparation at “high” level on “First Order Logic” definition.

3.2 Composition and canonical operators

The model describes component composition using an acyclic directed graph. Formally, composition is a graph \(G\) that consists of a set \(N\) of nodes and a set \(E\) of edges. Each node represents a component and each edge a temporal constraint between two nodes. Nodes representing dynamic components can be annotated by a \(\text{Setting}\); in this way, component creator can define values for a subset of \(\text{Input}\) of the component.

The graph (called conceptual map) defines a partial order relation among nodes (themes developed by component) and from the point of view of authors is a good representation for contents of composed
components. The composition graph will be adapted during the delivering process to obtain a delivering graph matching goals and profile to a particular learner (see section 4).

Operators sequence, alternate and parallel allow us to express composition. They can be combined recursively.

(a) Sequence: Coid, SEQ Coid, learner has to traverse components identified by Coid and Coid. Coid can be accessed only if Coid, is accessed successfully.
(b) Parallel: Coid, PAR Coid, learner has to traverse components identified by Coid, and Coid. Learner can access Coid, and Coid, independently; there is no order relation between them.
(c) Alternate: Coid, ALT Coid, learner has to traverse components identified by Coid or exclusive Coid. At learning time, the system will choose between Coid, and Coid, according to a particular UM.

Table 1. Rules used building Input-Output for composed component.

<table>
<thead>
<tr>
<th>Coid in</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, OPE C</td>
<td>{I / l \in (Input-C_i - setting-C_i) }  k = i or k = j</td>
<td>{O / O_i \in Output-C_i } k = i or k = j</td>
</tr>
<tr>
<td>C, ALT C</td>
<td>G_i = {I / l \in (Input-C_i - setting-C_i) }</td>
<td>G_1 = {O / O_i \in Output-C_i }</td>
</tr>
<tr>
<td></td>
<td>G_2 = {I / l \in (Input-C_i - setting-C_i) }</td>
<td>G_2 = {O / O_i \in Output-C_i }</td>
</tr>
</tbody>
</table>

Table 2. Rules used building Contents for composed component.

<table>
<thead>
<tr>
<th>Coid in</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, OPE C</td>
<td>{&lt;C_m, role_m, concept&gt;, \in {R_m} }  k = i or k = j</td>
</tr>
<tr>
<td>C, ALT C</td>
<td>G_1 = {&lt;C_m, role_m, concept&gt;, \in {R_m} } of C_1</td>
</tr>
<tr>
<td></td>
<td>G_2 = {&lt;C_m, role_m, concept&gt;, \in {R_m} } of C_j</td>
</tr>
</tbody>
</table>

Table 3. Rules used building Prerequisites for composed component.

<table>
<thead>
<tr>
<th>Coid in</th>
<th>Prerequisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>C, SEQ C</td>
<td>{&lt;C_m, role_m, concept&gt;, knowledge-level }, \in {R_p} } of C_k  k = i or k = j AND ( {&lt;C_i, role_m, concept&gt;, \in {R_p} } of C_k )</td>
</tr>
<tr>
<td>C, PAR C</td>
<td>{&lt;C_m, role_m, concept&gt;, knowledge-level }, \in {R_p} } of C_k  k = i or k = j AND ( {&lt;C_i, role_m, concept&gt;, \in {R_p} } of contents-C_m )</td>
</tr>
<tr>
<td>C, ALT C</td>
<td>G_1 = {&lt;C_m, role_m, concept&gt;, knowledge-level }, \in {R_p} } of C_k</td>
</tr>
<tr>
<td></td>
<td>G_2 = {&lt;C_m, role_m, concept&gt;, knowledge-level }, \in {R_p} } of C_j</td>
</tr>
</tbody>
</table>

Example 2: The expression C_{10} = C_1 SEQ (C_3 PAR (C_2 SEQ (C_5, ALT C_4))) defines the component shown in Fig. 3. Note that from now and by simplicity we will use C_i on operators meaning Coid_i. This composition graph denotes two delivering graphs, G_1 and G_2.

For a composed component, its composition provides information to process automatically almost all its metadata. Tables 1, 2 and 3 describe rules used to define them. In these tables, OPE means operators SEQ or PAR.

In Table 1, Input of C_m = C_i SEQ C_j is the union of inputs of both components excluding I's that have been set by creator of C_m. In table 2, Contents of C_m = C_i ALT C_j is equivalent to contents of Coid_i or exclusive contents of Coid_j. Whereas, in table 3, Prerequisites for C_m = C_i PAR C_j includes all prerequisites of Coid_i and Coid_j, except concepts developed by them. On the other hand, Acquisition Function for composed component can be defined from acquisition functions defined for C and C_j (for instance, by function composition). Finally, note that binary operators (SEQ, ALT, and PAR) can be easily generalized for n components.

Figure 3. Example of composition.
3.3 High-level operators

In certain cases, authors would like to merge components or fragments of them to build a new component. Canonical operators cannot handle this mix of components; in consequence, we propose two high level operators:

a) **Aggregation:** \( C_k = C_i \text{ AGG } (C_i.C_a \text{ SEQ } C_j.C_b) \). AGG mixes two components adding the arc \(<C_a,C_b>\) for coalescing their compositions. \( C_i.C_a \) must be always a node without successor and \( C_j.C_b \) must be always a node without predecessor. (See table 4 defining the new composition).

b) **Projection:** \( C_k = C_i \text{ PROJ } (\text{Composition-Expression}) \). PROJ builds component \( C_k \) from a subset of the composition of \( C_i \). Composition expression is built using SEQ, PAR and ALT. See table 4 defining the new composition.

To calculate metadata for both operators, we can use rules defined in Tables 1, 2 and 3.

<table>
<thead>
<tr>
<th>Table 4. Composition for aggregation and projection operators.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_i = C_j \text{ AGG } C_k ) (Composition-Expression)</td>
</tr>
<tr>
<td>( C_i.C_a \text{ SEQ } C_j.C_b )</td>
</tr>
</tbody>
</table>

**Example 3.** Let us again turn our attention on component \( C_{10} \) (Example 2). Let suppose \( C_{16} = (C_8 \text{ SEQ } C_9) \). The expression \( C_{17} = (C_{10} \text{ PROJ } (C_2 \text{ SEQ } (C_3 \text{ ALT } C_4)) \text{ AGG } C_{16}) \) (\( C_{16} \)) takes a sub-composition from Coid{10} that defines an order relation between \( C_{10}, C_4 \) and \( C_8, C_9 \) and coalesces the sub-composition with component \( C_{10} \) to build \( C_{17} \). Figure 5 visualizes composition of component \( C_{17} \).

3.4 Failure handling

We introduce failure notion to explain what happen when a learner does not acquire a concept surfing on composition.

3.4.1 Definition

A component is used unsuccessfully when its acquisition function returns FAIL. It is denoted by the expression \( \neg C_i \). In consequence, we define the following failure expressions:

a) \( \neg C_i \text{ SEQ } C_j \) : that is to say, after a failure of \( C_i \) try \( C_j \)

b) \( \neg C_i \text{ SEQ } C_j^{(n)} \) : it means that after a failure of \( C_i \) try again at most \( n \) times.

c) \( \neg C_i \text{ SEQ } \text{ FAIL} \) : that is to say, after a failure of \( C_i \) propagates the failure to the overall component.

3.4.2 Failure handling semantic

There are two implicit rules to handle failure semantic i) if a failure processing return FAIL, it enforces the failure of the overall component ii) if a failure processing is successful, the processing of the overall component is continued as if there was no failure.

**Example 4.** The definition of the component \( C_{10} \) (see example 2) is extended to handle failure.

a) \( \neg C_i \text{ SEQ } C_j \) after a failure on \( C_i \) try \( C_j \). If \( C_i \) return FAIL, it propagates failure to \( C_{10} \).

b) \( \neg C_i \text{ SEQ } \text{ FAIL} \) after a failure of \( C_i \) propagates failure to \( C_{10} \).

c) \( \neg C_i \text{ SEQ } C_j^{(2)} \) after a failure on \( C_i \) try at most 2 times again; in other case, propagates failure to \( C_{10} \).

In some self-learning environment, learners could need to disable failure handling; components can provide an \( I_i \) like \( \langle \text{SetFailureHandling}, \{\text{value}\}, \text{possible-values (Enable, Disable)} \rangle \), where \( I_i \) is Input of component.
4. THE AUTHORING AND DELIVERING PROCESSES

The framework for our educational components will be an adaptive web-based learning system. Components will be deployed into development and runtime environment (i.e. at authoring time and at delivering time). At authoring time, we can identify the following main actions:

- **The addition of primitive components.** In this case, creators must define metadata for the new component. Note that to classify a component, they only need defining relationships with concepts from DM. Our approach contrasts with LOM standard, which proposes metadata "Relation" to define relationships among learning objects. In our system, components are completely independent each other; it will facilitate reusability. Besides, DM, Contents and Prerequisites allow us inferring relations among components.

- **The design of composed components.** Creator, using a G.U.I, can draw the conceptual map for a composed component, which can be translated later in terms of canonical operators. Our system calculates automatically almost all metadata for the new component from metadata of components inside its composition. The system also assists authors verifying consistence of the composed component. Authors can use different granularity level components, from components developing only a role for only one domain concept to component developing several concepts.

![Figure 6. our adaptive process](image)

*Figure 6 shows our adaptive process. At authoring time, authors specify a conceptual map (CM) of the learning objects, which denote a set of delivering graphs (DG). A delivering graph is a graph without operators ALT. This set of delivering graphs will be filtered at delivering time. The user preferences (e.g. the type of media, the language) are the first filter. After that, the system will select the delivering graph having prerequisites satisfied by the user model. If there are several graphs satisfying this step, the system will choose only one. If the resultant set is empty, it implies that the current user cannot access this course because he/she has not sufficient knowledge (the system can states the missing knowledge). Finally, the selected graph is simplified, that is all the nodes having their content already known by current user are annotated (see adaptive navigation and presentation (Brusilovsky, P. 1996)). Of course, if the same conceptual graph is delivered to another user, the selected delivering graph can be different.*

5. CONCLUSION

In this paper, we present an educational component model for reusable learning objects and describe briefly its framework, i.e. the adaptive web-based learning-system. The model is based on a semantic description of each component and can be viewed as an extension of classical metadata standard such as SCORM. Our claim is that semantic metadata are required to allow a real reusing and assembling of educational component.

We define an educational ontological, which is used to define each knowledge domain, semantic of the components, and user model. Components are described by metadata, which integrate their semantic (contents, pre-requisites, acquisition function), inputs, outputs and other characteristics. Composition is described by an acyclic directed graph, which powers sequencing and navigation capabilities of composed leaning objects. Elementary components can be combined using canonical constructors which have a formal
definition; whereas, high-level constructors allow us to coalesce and to copy components. Besides, composed components are classified automatically by system. A sophisticated authoring tool assists this process. A prototype of the authoring tools, supporting the component model, is under construction.

Our vision is that by each knowledge domain, a large repository of educational components will support adaptive learning. In consequence, the component model provides mechanisms to allow building a lesson on the fly, replacing of component by other at deployment time, supplying alternatives graphs with the same component and supporting link adaptation on the composition graph.

Future works will address several issues. We plan to extend our component model with intentional components. Intuitively, an intentional component is a composed component where a specific component (e.g C_10) is replaced by a searching expression on the component repository. These components will allow authors to express courses that are more abstracts which increasing reusability and adaptation. Finally, we plan to investigate into a more distributed vision of our system. At the moment, we only consider a central repository, but it will be interested to cover distributed repositories corresponding to different organizations.

REFERENCES


