# An Architecture for Candidacy in Adaptive eLearning Systems to Facilitate the Reuse of Learning Resources

Declan Dagger, Owen Conlan, Vincent P. Wade,
Trinity College Dublin,
Republic of Ireland.

**Email** {Declan.Dagger, Owen.Conlan, Vincent.Wade}@cs.tcd.ie

### *Abstract*

Because of the significant time, money and effort devoted to creating online learning resources, one of the key challenges of producing this learning experience is to reduce the costs involved with authoring, re-authoring and re-purposing learning resources, i.e. any digital media that aids in the process of learning [*Brusilovsky, P., Eklund, J., and Schwarz, E. (1998), De Bra, P., Aerts, A., Houben, G.J., Wu, H., (2000)*]. The situation often arises whereby the portability of a developed learning resource will not be realized as the learning resource is not marked up in a standardized way. By applying a standards-oriented approach to learning resource candidacy potentially increases the reusability of that learning resource. The methodology of candidacy is the abstract grouping of learning resources with similar goals, objectives, learning style and possibly any other grouping criteria that may arise. Implementing candidacy with the multi-model approach can produce not only reusable learning resources but also reusable and durable adaptive courses.

### *1 Overview*

This paper describes the current standards of eLearning. It presents an architecture for candidacy in adaptive eLearning systems to facilitate reusable learning resources and adaptive courses. The paper explains the design-time candidate creation process. It explains the process of selecting the appropriate candidate learning resource at runtime through the multi-model architecture. The paper then introduces the OAT (Open Authoring Toolkit) for Adaptive Courseware Construction which is used to produce Adaptive Courseware for the multi-model architecture in an author-supported environment.

### *2 Metadata and Standards*

This section will introduce eLearning standards and their associated metadata. It will focus on the main standard from the ADL (Advanced Distributed Learning) initiative.

#### 2.1 Metadata

Metadata can be defined literally as "data about data," but the term is normally understood to mean structured data about digital and non-digital resources that can be used to help support a wide range of operations. These might include, for example, resource description and discovery, the management of information resources (including rights management) and their long-term preservation. The main driving force behind metadata development and advancement is reusability, accessibility, interoperability and durability. Metadata is used to tag a resource with some low level descriptive information that can be intuitively interpreted. The core developers of eLearning-related metadata standards are ADL, IEEE LTSC, AICC, IMS Global Consortium, Dublin Core and ARIADNE.

#### 2.2 SCORM (Sharable Content Object Reference Model)

The ADL SCORM standard is the result of several standardization efforts of the Institute of Electrical and Electronic Engineers Learning Technology Standards Committee (IEEE LTSC), Instructional Management Systems (IMS) Global Learning Consortium, Dublin Core and the Aviation Industries Computer-based training Committee (AICC). SCORM extends the IEEE LTSC Learning Object Metadata (LOM), the IMS Learning Resource XML Binding Specification and Simple Sequencing Definition Model, the Dublin Core Metadata Initiative vocabularies and the AICC Computer Managed Instruction (CMI) data model.

SCA (Sharable Content Asset) is a new term introduced in SCORM version 1.3 applications profile working draft. With older versions of SCORM the learning resources that could not be tracked or directly

launched by the Learner Management System (LMS) were called assets. This caused some confusion over the meaning and implementation of SCORM Assets. SCA is defined as a launchable collection of assets. SCA differs from SCO (Sharable Content Object) in that it cannot communicate with the LMS via the API Adapter Applet. It does however have associated metadata for the purposes of resource discovery.

A SCO (Sharable Content Object) consists of one or more SCAs that are grouped to form a single learning resource that can be launched by the LMS and tracked by the API Adapter. A SCO represents the smallest runtime environment traceable unit. SCOs can be grouped together in hierarchical fashion to produce a learning experience. They also have associated metadata for discoverability and reusability purposes. Since the SCO communicates with the runtime environment it must implement the minimum number of API calls allowed, namely LMSInitialize() and LMSFinish().

Learning resources are grouped into logically coherent units called Content Aggregations (CA). These aggregations can be used to represent some taxonomic structure. The role of the CP is to connect the CA with their associated metadata and also group all necessary resources into one deliverable package. The CP (Content Package) is used to transfer a SCORM conformant learning experience from one SCORM conformant LMS to another SCORM conformant LMS.

SCORM's Sequencing Definition Model directly extends the IMS Simple Sequencing Definition Model. It can be used by course developers to specify certain sequencing behaviors. There are eleven definition categories available in this model and a binding specification which describes how the definition model values are bound to XML elements and attributes.

## *3 Candidate Content Groups (CCG)*

Traditionally Adaptive Hypermedia Systems (AHS) interact with only two models, the learner model and the content model *[Brusilovsky, P., Schwarz, E., Eklund, J. (1998); DeBra, P. Stash, N. (2002)]*. One problem of this approach is the coarse nature of the content model. The multi-model approach outlined in this paper can interact with and interpret N models to produce adaptivity. This section describes an abstraction technique, developed at Trinity College Dublin, for grouping similar learning resources into Candidate Content Groups (CCG).

### 3.1 Abstraction through Candidacy

The use of metadata to markup learning objects so that they can be searched for, retrieved and used can be seen in implementations such as SCORM, ARIADNE and EASEL [Conlan, O.; Hockemeyer, C.; Wade, V.; Albert, D.; Gargan, M. (2002)]. However just providing metadata which describes the learning resource does not necessarily achieve reuse. Learning resources need to be "right grained" for reuse, these metadata descriptions need a common vocabulary, common schema and common communication API. Some of these issues are addressed in the current learning resource metadata standards but there is still no standard approach to grouping sets of related learning resources.

Although the current standards use metadata to markup learning resources, one of the main restrictions of their approach is that the metadata is too low level to adequately capture the context semantics of the learning resource. From these limitations came the high-level and abstract approach of learning resource candidacy.

To facilitate flexibility in the design and implementation of new course offerings the multi-model approach was designed to include an abstraction mechanism. This mechanism enables the design-time collaboration of many people on the development of an adaptive course. For example, the abstraction enables the course author (knowledge domain expert) to develop a teaching strategy describing the course sequencing not in terms of the learning resources to be added, but in terms of the concepts to be learned. This abstraction allows the course author and instructional designer to design the course in a more structured way without necessarily being concerned with the individual pieces of content that will be used to populate the final course.

This abstraction is facilitated through candidate groups. Candidate groups are used to group together *like* learning resources. For example, a candidate content group concerned with a particular learning concept may contain several learning resources, each covering the learning concept in different ways. In the case of learning resources these differences may be pedagogical or technical – some learning resources may deal with the concept from different perspectives or render the material differently for different network topologies (Peer-To-Peer, Wireless or Fixed), bandwidth availability situations (modem or 10/100 Mbps E1), end-user devices (mobile phone, PDA, tablet PC, laptop or desktop) and different learner abilities (special input devices for handicapped learners).

Each candidate group has associated metadata that describes the role of the group and has the identifiers of the constituent learning resources within the group. A simple candidate content group may be visualized as –

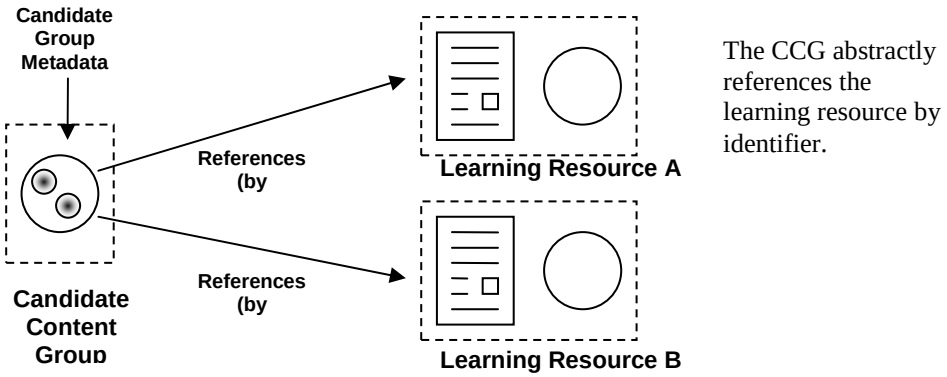



**Fig 1) Candidate Content Group**

In the figure above, the candidate content group has two candidates, A and B. Learning resources in the same candidate content group are equivalent on some axis, usually the concept they teach. Candidate groups can be formed for any set of models in the multi-model system, content, narrative, etc. For example, there can be candidate teaching strategy groups containing teaching strategies that produce equivalent courses according to different instructional approaches.

As candidate groups refer to their candidates by identifier it is possible for any single model in the system to be included in multiple groups. Groups do not have to be homogeneous and may contain models of different types. For example, a candidate group may have some teaching strategy and some learning resource candidates.

For example, a Candidate Content Group (CCG) could be created to introduce the learner to the concept of database data types as illustrated below. The CCG would identify the learning resources that can potentially teach the required concept, in this case database data types, and group them together with some descriptive information. The members of the CCG, mod052-000 and mod052-000b, are abstract references to the physical learning resources. The adaptivity type, *db.datatypes.introduction*, identifies the competency taught by the CCG, i.e. database data types. This CCG was created to group *like* learning resources by the competencies with which they teach. There are many methodologies that can be applied to the grouping of learning resources, such as learning style similarity, prerequisite knowledge similarity and suitable end terminal similarity. The abstract information provided within these groups can be inferred by the adaptive engine during the candidate selection process to render the appropriate learning resource. This abstract information which describes the educational context in which the learning resources are being used is captured in a meta-model called MetaSCO.

```
<?xml version="1.0" ?>
<candidategroup>
  <general>
    <identifier>cgmod052-000</identifier>
    <description>This CCG groups together the learning resources that can be used to introduce
      the concept of database datatypes.</description>
  </general>
  <members>
    <member>mod052-000</member>
    <member>mod052-000b</member>
  </members>
  <educational>
    <adaptivity>
      <adaptivitytype name="competencies.taught">
        <set type="ALL">
          <candidate>
            <langstring lang="en">db.datatypes.introduction</langstring>
          </candidate>
        </set>
      </adaptivitytype>
    </adaptivity>
  </educational>
</candidategroup>
```

**Fig 2) Example metadata of a Candidate Content Group**

### 3.2 MetaSCO

Candidate Content Groups are used to abstractly group learning resources that share a common goal, objective, competency or prerequisite. CCG aid the adaptive engine in selecting the appropriate candidate to be delivered. MetaSCO describes the contextual relationship between concept and learning resource (SCO). It is the capture and embodiment of the context of the prior usage of the learning resource to teach a particular concept. The information contained in a MetaSCO can aid the course developer to identify potential candidate learning resources. A MetaSCO may reference many learning resources and a learning resource may be referenced by many MetaSCO.

MetaSCO is another form of XML metadata that can be stored in a learning resource repository. One research area in artificial intelligence that can aid in the storing of such MetaSCO is clustering. Clustering involves the grouping of similar objects, for example learning resources, from a given set of inputs, for example encapsulated abstract information about the prior usage of the learning resource. Clustering of MetaSCOs can provide the course developer with a mechanism to perform a high-level concept-based search across the learning resource repository.

With the growth and adoption of semantic web, the key challenge of creating globally accessible, interoperable and reusable learning resource repositories is a common and standardized approach to communicating with the repository. Since this methodology for communicating with the learning resource repositories does not yet exist, semantic web techniques must bridge the gap. For each learning resource repository there should exist an accessible ontology or hierarchy of ontologys that contain structured information about the contained learning resources. This ontology can be used to access and interact with the learning resource repository. MetaSCO is a mechanism for capturing the context-based prior usage of the learning resource. It can be used during the construction of the learning resource repository's ontology to provide a more rich description of the learning resources contained within the repository.

## *4 Candidate Selection*

This section introduces the execution models used to select the appropriate learning resource from a candidate content group. It describes the narrative model, the candidate selector and the narrative execution process.

### 4.1 Narrative Model

The narrative model is the encapsulation of the expert's knowledge of a domain of information. The narrative model captures the logic behind the selection and delivery of a learning resource within the scope of an adaptive course. The narrative model allows the course author to separate the intelligence, which performs the adaptivity, from the content. This separation increases the potential for the reuse of the learning resources involved, i.e. the content and the intelligence.

Narratives can be used to generate adaptive course offerings that differ in ethos, learning goals, pedagogical or andragogical approach and learner prior experience. These offerings can be made from single or multiple learning resource repositories. The vocabulary used to describe the learning concepts embodied in the course offering is that of the domain expert. As the narrative does not refer directly to individual learning resources, but rather to candidate content groups using this vocabulary, the domain expert can create the narrative without being constrained by pedagogical or technical delivery issues at the content level. The author can simply refer to the candidate content group in the narrative and allow the adaptive engine determine which candidate from the group is most suitable for delivery.

The primary goal of the narrative is to produce courses that are structured coherently and fulfill the learning goals for the adaptive course in a way that engages the learner. It is, therefore, the domain expert's task to ensure that each learning goal has sufficient learning concepts to fulfill that goal and that those concepts are sequenced appropriately.

From this perspective the domain expert must consider how the exclusion or inclusion of concepts or sequences of concepts, in the case of sub-narratives, will impact on the intelligibility of neighboring concepts and on the personalized course as a whole. To this end it is often useful to determine, before designing a narrative, what is the granularity of personalization that is to be achieved, i.e. personalization on the section, page or paragraph level. This decision is influenced by both the granularity of the vocabulary describing the concepts and the granularity of the learning resources that will fulfill that concept. The granularity of personalization cannot be smaller than the larger of the vocabulary or learning resource granularity.

When designing a narrative it is also useful to determine what learning resources are considered to be potential core material and are always present in all personalized offerings. With granularity of personalization and the core learning resources determined the expert has a framework in which to consider the impact of the inclusion or exclusion of concepts based on the learners' expertise and preferences. The structure in which the learning resources are placed should be completely open and defined by the course author in the narrative. This enables the course author to produce courses from any pedagogical model they desire and not be constrained by a static system course model.

```
)

(dom "set-root" "course")
(dom "add-to-parent" "learnerid" (get-learnerid))
(dom "add-to-parent" "title" "SQL Course")

(dom "add-to-parent" "section")
(dom "add-to-parent" "name" "Data Types")
(dom "add-to-parent" "page")
(add-element "pagelet" (select-learningstyle-candidate (select-competencies-candidate "cgmod051-000")))
(dom "set-parent" "..")
(dom "add-to-parent" "page")
(add-element "pagelet" (select-learningstyle-candidate (select-competencies-candidate "cgmod052-000")))
(add-element "pagelet" (select-learningstyle-candidate (select-competencies-candidate "cgmod052-000b")))
(add-element "pagelet" (select-learningstyle-candidate (select-competencies-candidate "cgmod052-001")))
(dom "set-parent" "..")
```

The narrative is interpreted to create the personalized adaptive course

**Fig 3) Sample Narrative.**

### 4.2 Candidate Selector

Closely related to narratives, or more precisely sub-narratives, are candidate selectors. Candidate selectors are the rule sets that choose a candidate from a candidate content group. Candidate selectors are constituted of a rule set and metadata describing the selector. When the execution of a narrative (or sub-narrative) meets a candidate group, a candidate selector is executed. Within the system there may be several candidate selectors available for choosing candidates for different purposes. For example, there may be a candidate selector for selecting candidates based on the screen size of the delivery device.

Candidate selectors may be executed in two modes – **all** and **best**. The all mode returns a set of candidates that all meet the minimum requirements of the selector. The best mode returns a single candidate that best fits the requirements. Using these modes it is possible to call several candidate selectors on a candidate group – each refining the selection until the final candidate selector is asked to make a best selection. For example, a candidate group may contain pagelets covering a concept with different pedagogical approaches and for different devices. There should be two candidate selectors (one for each variable axis) – one to

select **all** appropriate candidates for the device, the second to select the **best** candidate, from the subset, according to the learner's learning preference. If the learner is a predominantly visual learner and is using a PDA to view the course the first candidate selector will select all learning resources suitable for a PDA and the second will choose the learning resource most appropriate for visual learners.

Each candidate selector may have descriptive metadata associated with it. This facilitates the reuse of the candidate selectors by other course authors and aids in the creation of the learning resource repository. The metadata may describe the criteria the selector uses and the algorithmic basis for that selection. For example, a candidate selector's metadata may state that is uses Pearson's Correlation to perform the numeric comparison of learner preference values to learning resources metadata values

```
(deffunction select-competencies-candidate (?name)
        (if (eq (compare-attrib "competencies" (get-learnerid) ?name) "TRUE")
                then
                        (return ?name)
                else
                        (return "NOT REQUIRED")
        )
)

(deffunction select-learningstyle-candidate (?name)
        (if (eq ?name "NOT REQUIRED")
                then
                        (return ?name)
                else
                        ; Get the candidates identifiers from the candidate group
                        (bind ?candidates (get-content "subcollection4" ?name "//*[name()='member']"))
                        ; Use a candidate selector to select the best candidate
                        (bind ?successful-candidate (candidate-selector "learningstyle.kolb" ?candidates))
                        (return ?successful-candidate)
        )
)

(deffunction add-element (?elementname ?content)
        (if (eq ?content "NOT REQUIRED")
                then
                        (+ 1 1)
                else
                        (dom "add-to-parent" ?elementname ?content)
        )
)
```

The candidate selectors are used to select all appropriate candidates from a candidate content group.

**Fig 4) Sample Candidate Selector for Competencies and Learning Styles**

### 4.3 Narrative Execution

Narratives and sub-narratives are rule sets that are executed in the rule engine. Every adaptive course has at least one narrative. The main narrative is referred to as the root narrative. It is the role of the root narrative and any children narratives to produce a personalized course offering tailored to the learner model and any other models they wish to access. In this sense it is the narrative model(s) that reconcile the other model information to produce the personalized course.

In order to provide as much functionality as possible to the course author/instructional designer the rule language expressed in the narratives and interpreted by the rule engine should be as rich as possible. It is envisaged that the course author would design new courses using a graphical interface with design support facilities, but if the author wished to create narrative structures/sequences manually based on more complex algorithms they should not be limited by the rules language.

As the narrative decisions are based on model information the rule engine is able to access any model or candidate group information when executing narratives. This information is used to influence decisions made in the narratives. As candidate selectors are based upon the same rule language as narratives and executed by the rule engine they also have access to any model information they require to make a selection. For example, a narrative that adds concepts based on the learner's prior knowledge would, through the rule engine, look at the learner model repository to access the current learner's model. It would then query the learned competencies of that learner before adding the concept. If that concept was represented by a candidate content group with several learning resources the rule engine would execute a candidate selector to choose the appropriate learning resource. The candidate selector would use the rule engine to access the content model of each candidate pagelet before making its selection.

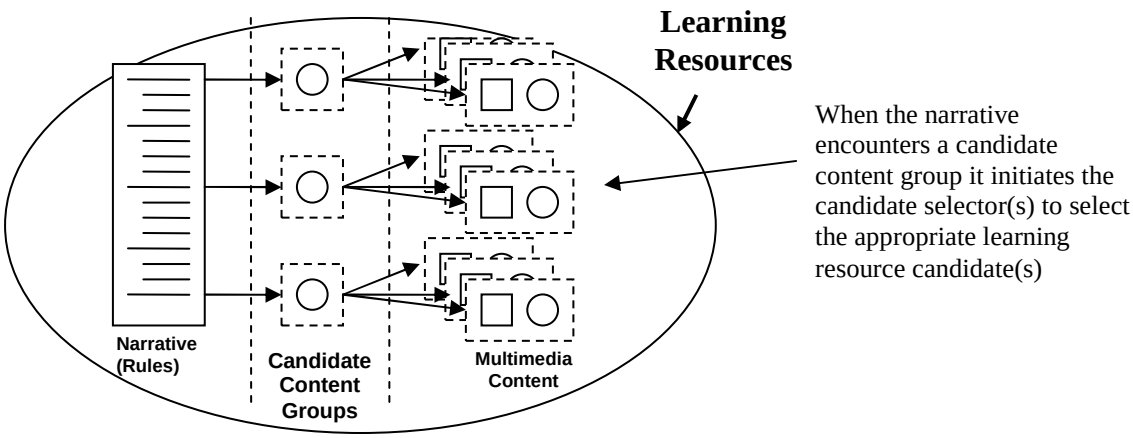



**Fig 5) Execution flow of a Narrative**

The ultimate output of the Adaptive Engine is a personalized course model. The structure of this course should, however, be open allowing the narrative author to determine the structure. In this way the author does not have to comply with a restrictive course model dictated by the Adaptive Engine. For example, a course author may wish to have a course-section-unit-page taxonomy in an adaptive course. The creation of this, or any other structure, is possible through the narrative.

## *5 Architecture for Candidacy*

This section describes the multi-model architecture for candidacy. It introduces the OAT system that can create adaptive courses for delivery on the multi-model architecture.

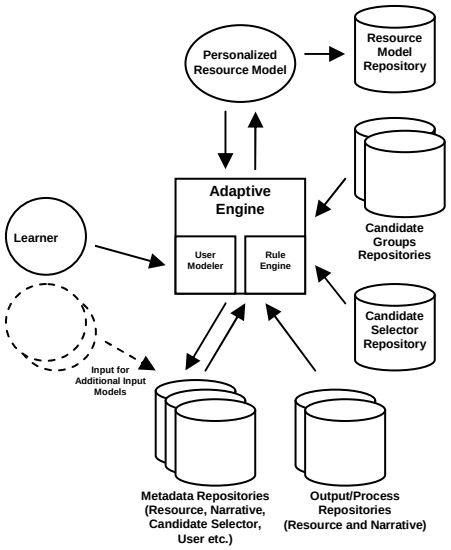
### 5.1 Multi-Model Approach

Traditional development methods of Adaptive Hypermedia Systems describes many diverse approaches to implementing AHSs but these approaches have one common feature that limits the reusability of their learning resources. This limitation stems from the design approach taken in these systems whereby the learning resource and logic for producing adaptive effects are intertwined. By merging the learning resource and logic, these systems limit the reusability of that learning resource as the embedded logic often has dependencies on other learning resources or requires a context dependent engine to execute its rules logic.

Such embedded logic also restricts system developers as they must have a complete knowledge of the possible outputs of the rule system and how this new resource (and embedded logic) will impact on the service execution. This requirement for complete knowledge of the system and its execution logic also limits the possibilities for collaboration and reduces ease of extensibility between system developers (knowledge domain experts), resource designers and instructional designers (pedagogical experts) as if any modifications have to be made to the system they must each have a complete overview of how the resource and sequencing logic interacts.

Considering these limitations on traditional approaches to designing AHSs the multi-model approach has been designed to –

- Divide the resource and the sequencing logic into separate extensible models.
- Enable the integration of additional models to support the additional axes of adaptivity.
- Enables easier collaboration by system designer, resource designer and instructional designer at an appropriate level of abstraction.
- Utilize a flexible metadata driven approach to aid reuse of models.




The adaptive engine (AE) interacts with the learner to build a personalized learner model.

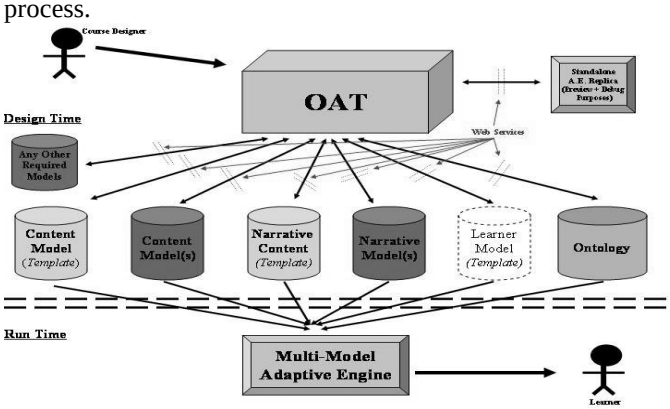
The AE interacts with and interprets the learner model, the content model, the narrative model, the candidate content groups and any other required models to produce the adaptive personalized course model for the learner.

This personalized course model is transformed to create the personalized course for the learner.

**Fig 6) General Multi-Model Architecture**

### 5.2 OAT (Open Authoring Toolkit) for Adaptive Courseware.

The process of creating an adaptive course can be time consuming. The OAT is a service-based toolkit created to automate the process of creating an adaptive course and provide a GUI to aid the course author in the authoring process. The OAT offers support to the course author by automating the model markup process.




The OAT can interact with, interpret and create N Models.

**Fig 7) OAT Architecture**

The OAT toolkit can be used by the adaptive course developer to select appropriate teaching strategies. The author can build multiple concept spaces which represent the concepts to be taught by the adaptive course. The OAT can search across multiple learning resource repositories by keyword and by prior learning context. The OAT can automatically create Candidate Content Group and MetaSCO metadata. The tool can interact with template narrative models and create custom narratives. It can produce personalized adaptive courses for the multi-model architecture and has a built in test and debug engine for real-time evaluation of the created adaptive course. The tool can output the personalized adaptive course as a content package for portability purposes.

### *Conclusion*

This paper introduces an architecture for candidacy in adaptive eLearning systems to facility the reuse of learning resources. It also introduces the OAT which is a toolkit for creating adaptive eLearning courses.

### *References*

*[Brusilovsky, P., Eklund, J., and Schwarz, E. (1998)* Web-based education for all: A tool for developing adaptive courseware. *Computer Networks and ISDN Systems (Proceedings of Seventh International World Wide Web Conference, 14-18 April 1998, 30 (1-7), 291-300]*

*[De Bra, P., Aerts, A., Houben, G.J., Wu, H (2000)* Making General-Purpose Adaptive Hypermedia Work. *Proceedings of the WebNet Conference, pp. 117-123, 2000]*

*[Conlan, O.; Hockemeyer, C.; Wade, V.; Albert, D.; Gargan, M. (2002)* An Architecture for integrating Adaptive Hypermedia Service with Open Learning Environments. *ED-MEDIA 2002, World Conference on Educational Multimedia, Hypermedia & Telecommunications, Denver, Colorado, June 2002]*

*[DeBra, P. Stash, N. (2002)* AHA! Adaptive Hypermedia for All. *Proceedings of the SANE 2002 Conference, Maastricht, May 2002, pp. 411-412]*