Using the SCORM Standard to Build Adaptive Content Packages in RELOAD

Pierpaolo Di Bitonto¹, Teresa Roselli¹, Veronica Rossano¹, Cristina De Serio
¹Department of Computer Science,
University of Bari “Aldo Moro”
Bari, Italy
{dibitonto, roselli, rossano}@di.uniba.it

Abstract—It is universally recognized that learning is more effective if the didactic content is tailored to the specific user’s needs. Since the advent of Adaptive Hypermedia, research in e-learning has always investigated, from a theoretical point of view, the different dimensions involved, and from a technological point of view, the tools helping to integrate more adaptive contents. The problem is that these are often ad hoc solutions which are not interoperable at all. This paper presents a solution that aims at using the SCORM standard in order to build an adaptive content package. The strength of this work is that it applies the adaptation model using some API functions which affect only the client side of the package, without affecting the standard and the interoperability with the LMS. The paper also presents some additional functionalities for the RELOAD authoring environment intended to support non-expert users in building adaptive content packages.

I. INTRODUCTION

The spread of e-learning has generated a profound modification of learning paradigms, opening out new pathways requiring new procedures and tools. One of the most interesting challenges in e-learning research is to define new models serving to adapt learning contents to the specific learner’s requirements. It is universally recognised, in fact, that personalisation of the content improves the effectiveness of the learning process, because the resources proposed to the student will be in the form best suited to her/his requirements [2]. In particular, the most commonly used approach in adaptive learning research uses the student’s learning styles as a basis on which to adapt the content [3, 4, 9, 10]. Moreover, the use of technologies in educational contexts allows the navigational path to be adapted during the user interaction, thus improving the student’s learning experience and tailoring it more closely to each student’s characteristics. Based on these claims the present research, conducted in collaboration with a research unit in cognitive psychology led by Maria Sinatra, professor at the University of Bari, has allowed us to define an adaptation model for multimedia learning content based on different students’ cognitive styles, already presented in [8]. One of the main problems in defining an adaptation model is its interoperability. The majority of the proposed solutions, in fact, are ad hoc solutions that cannot easily be used in a different context. The main purposes of this research were, on one hand to define an adaptation model which allows the content to be tailored to the user’s cognitive styles, and, on the other, to investigate whether it is possible to build a SCORM compliant adaptive content package. The paper closely examines the technical aspects faced when using the SCORM standard to build adaptive learning contents. The main aim of the research is to apply the previously defined theoretical adaptation model [8] using a standard technology. It is clear that this should not involve an extension of the specifics, that would hamper the interoperability of the standard, but would focus on an attempt to reuse them to achieve the adaptation process, according to the defined model. The research was then developed with the design of some solutions to be integrated in one of the authoring environments most commonly used to create adaptive content packages, RELOAD [12], in order to support non expert users. Thus, the research questions posed were: is it possible to build a SCORM compliant adaptive Content Package? Is it possible to facilitate the building process of adaptive content packages? Are there any authoring tools available that can do this? The paper addresses these questions in the following sections: in section II the adaptation model according to the student’s cognitive styles is described; in section III the adaptive content package is presented and some limitations of SCORM are discussed; section IV describes the technical details of the SCORM standard; section V proposes some functionalities that could be added to one of the most popular authoring tools. Finally, some conclusions and future research directions are outlined.

II. THE ADAPTATION MODEL

It is widely recognized that the learning performance is improved if the learning content is adapted and tailored to the learner's cognitive style [3, 5, 13, 14, 15]. For this reason, in this work an adaptation model for multimedia didactic resources has been defined, in collaboration with Professor Maria Sinatra’s research group on cognitive psychology. The starting point of the adaptation process, detailed in [8], is defining the different cognitive styles according to a process that aims at discovering which presentation modes the specific student prefers. The questionnaire used is the Italian Cognitive Styles Questionnaire (ICSQ) defined by De Beni, Moè, and Cornoldi [7]. The ICSQ consists of a series of questions that are posed to assess whether the subject prefers the “wholist” or the “analytic” style, or else the “verbalizer” or “imager” styles. The basic idea underlying the defined adaptation model is that
each subject may prefer a particular presentation mode but that this preference can change during the interaction with the content. Thus, the result of the questionnaire is an ordered list of the student’s cognitive style preferences that is employed during use of the didactic content to establish which presentation style should be used with the student at that time. During the interaction, in fact, some test points are set and if the student passes the test the same cognitive style will be used for the next content, whereas if the student fails the test reinforcement content will be supplied. This step aims at reducing the probability that failure of the test is due to a difficult content or to lack of attention by the student. After the reinforcement has been given, the student has to pass a second test: if she/he is successful this time the same cognitive style will be used, otherwise the next cognitive style will be read (from the student’s profile) and used for the following didactic units. The model has been implemented using the learning content of the course on the Psychology of Communication and was experimented in two different degree courses: Informatics and Digital Communication, and the Humanities. The pilot study has demonstrated that the customization of learning paths according to students’ different cognitive styles promotes academic success [8]. In addition, the model has been implemented in a mobile application, named MoMaT (Mobile Museum Adaptive tour), which is aimed at guiding a visitor on a virtual tour of a museum [6].

III. THE ADAPTIVE CONTENT PACKAGE

The paper is focused on the SCORM standard, that was chosen with the intent to build an interoperable LO that could be easily integrated into any e-learning environment. The key concept of the model is to offer a high level of personalization, but in order to obtain this a high level of granularity is needed. Therefore, the didactic objectives need to be divided into different topics (Figure 2). For each topic two kinds of SCOs are built: one containing the learning content and one containing the reinforcement. Both of them contain use presentation modes suited to each cognitive style, and for both of them there will be a SCO which will contain the Comprehension Test (CT) to assess the student’s learning level. The CTs allow the content to be adapted according to the adaptation model described above. In order to make the model scalable and general, another type of SCO has been added to the theoretical model defined in the previous section. In fact, a cognitive style test SCO has been added, which contains the questionnaire serving to define the user’s cognitive style. In this way, the adaptive content package is sufficiently general to be applied independently of the cognitive style questionnaire used. The SCOs are then organised using a tree aggregation form that represents the domain organisation and the adaptation model described.

![Figure 1. The adaptation model](image)

![Figure 2. The tree organisation of the didactic content](image)

A. Limitations of the SCORM standard

The next part of the research was focused on the SCORM standard, with the aim of defining an easy procedure for building an adaptive content package. At first sight, an easy assumption is that Sequencing and Navigation rules would be sufficient to implement the defined adaptation model. However, a deeper study of the standard pointed out some limitations, so special solutions had to be adopted.

First of all, the preconditions for each content, defined on the basis of the goal assessment, allow an item to be skipped or hidden. In the adaptive model, instead, it was necessary to modify the sequence of the activities and the item to be shown according to the preconditions. Moreover, even though the standard claims that for each item different conditions could be defined (from 1 to n), during the implementation of the model it was found that more than two conditions caused problems in their evaluation. Another limitation is that the SCORM is not able to store information in a global variable that is common to all the items in the whole learning path. The only way to do this is to store it as a note for the learner, but this does not allow the information to be processed in run time. Moreover, in order to allow non expert users to build a SCORM compliant content package, an authoring tool is needed. This paper addresses all these limitations.

IV. REVISION OF THE SCORM STANDARD

The defined adaptation model has been developed using the SCORM 2004 standard 4th edition [1]. A study of the literature in this field revealed some researches aimed at adapting the functionalities offered by the SCORM [4, 11]. These researches aim at proposing extensions of the standard by creating a new version of the SCORM but this is not always compliant with the ADL version. As a consequence of this trend, the increasing number of personalised versions of the
standard causes a loss of interoperability between LOs and LMSs. For the sake of preserving this interoperability our research was focused on the client side of the content package navigation, leaving the standard unaltered. This is the main strength of the present research work, that extends the functionalities of the standard without changing its interoperability.

A. SCORM implementation details

In a SCORM compliant LO it is possible to distinguish two main logical components: the content and the navigational path. The SCORM standard in the CAM (Content Aggregation Model) discusses how the content should be organised and the SN (Sequencing and Navigation) how the user may navigate it. Both information items have to be included in the content package file Manifest in order to allow the content to be launched by the LMS. The navigational path within a LO is defined according to the learner’s activities, which are controlled by the RTE (Run Time Environment) and managed using Javascript API, which allows interaction between the LO and the LMS. To see how this research uses the SCORM standard to build an adaptive content package, while preserving the LO interoperability, the main SCORM elements need to be examined in more detail. Moreover, section II includes a detailed list of the API functions defined in the research in order to implement the adaptation model.

1) CAM Elements
The CAM elements describe the content organisation. The items serving in this research are:

- Organization: this contains the whole structure of the course. This element was important to define the control mode in the learning path;
- Item: this indicates that a node belongs to a specific Organization and identifies a SCO. The attribute identifier allows the resource to be identified. The value is used to define which resource should be associated to the item during the user interaction;

2) Sequencing and Navigation Elements
The SN elements in the file Manifest used to define the navigational path in the content structure are:

- adlnav:hideLMSUI: this allows the common navigational buttons (next and previous) of the LMS user interface to be hidden. The element has been used in order to define the rules that inhibit some navigational functions and to implement navigational actions among the SCOs according to the user’s interaction;
- imsxs:sequencingRules: is a container of sequencing rules. Each rule describes the behavior that each single object should have. The standard establishes that each item can have an unlimited number of rules. However, using the ADL Learning Management System, usually used to test the content packages, it was found that it is not able to manage more than two rules for each item. For these reasons an API function has been implemented. Moreover, the element imsxs:sequencing allows navigational rules among single SCO to be defined. In the case of adaptive content packages, rules have been defined in order to navigate among the nodes of the hierarchy according to the results of the Comprehension Tests;
- imsxs:objectives: is a parent element that contains all the learning goals of a specified activity (imsxs:objective). It includes both the primary objective and secondary objectives, if there are any. The primary objective element identifies the objective that contributes to the rollup associated with the current item, starting from the Sequencing rules of other items. The secondary objectives allow other items belonging to the same learning path to be evaluated in order to modify the path according to the pre and post condition verification. For example, let us suppose that after a first test to assess the learner’s knowledge the learning path could go two different ways: SCO A (containing the first lesson of the course) and SCO B (containing the lesson on the requisite prior knowledge for the course). In this case, the item that contains the SCO A will control, through a secondary objective “test”, that for the item with the primary objective “test” a sufficient score has been achieved to allow the SCO A to be supplied, otherwise SCO B will be supplied;
- imsxs:MapInfo: defines the mapping of an activity’s local objective information to and from a shared global objective. This information is important in order to be able to use a specific item as the reference for the learning goal of another SCO;
- imsxs:PreConditionRule: are rules that allow a specific SCO to be visualised or hidden. Those preconditions are evaluated by the LMS before the Initialize instruction is called. In the adaptive Content Package these rules are important in order to visualize the right SCOs according to the student’s learning style;
- imsxs:ruleConditions: is the container for the set of conditions to be applied as pre-condition, post-condition and exit condition rules. In other words, it contains a set of elements named imsxs:ruleCondition, which contain the conditions to be evaluated;
- imsxs:PostConditionRule: these are rules applied when an attempt is made to terminate the activity. In the case of the adaptive package the post conditions are important in order to control the navigational path after the CT has been completed. In order to show an item in a unit that has just been finished (if the user failed the test) the post condition rule modifies the control flow, going back on the tree activities and proposing the same didactic unit in a different presentation mode. Otherwise, if the user passed the test then the control flow goes on with the tree activities;
- imsxs:RuleAction: represents the intended action or behavior that the package should have after the evaluation of a set of specific condition sequencing rules. The attribute action represents the desired
sequencing behavior (skip, disabled, hiddenFromChoice, etc.) if the rule condition is evaluated as true;

- imss:ControlMode: is the container for the sequencing control mode information including descriptions of the types of sequencing behaviors specified for an activity. In other words, it allows the content developer to define how the navigation requests should be applied to an item and how the items should be considered while processing sequencing requests. The attributes used are: Choice, that indicates that a choice of navigation request is permitted (True or False) to target the child of the activity; choiceExit indicates whether the user is permitted to terminate the activity (True or False) if a navigation request is processed; Flow indicates that the control flow can go on (True or False) to the child of the activity; forwardOnly indicates that backward targets (in terms of the Activity Tree) are not permitted (True or False) by the child of this activity. Moreover, it is important to notice that in the adaptive content package, the Previous button has been hidden in order to prevent the learner from navigating freely in the learning content;

3) Run-Time Environment elements

The elements that have to be implemented in order to build the adaptive content package through API functions are:

- cmi.score.scaled: this indicates the learner’s performance. The value is used to define the navigational path in the tree activities;
- cmi.completion_status: indicates whether the learner has completed the SCO. Since the SCORM does not limit the ways to complete an activity, in the adaptive content package it will have different values (complete or not complete) according to the type of SCO (that can be CT, content, reinforcement or a cognitive style test)
- cmi.success_status: indicates whether the learner has mastered the SCO. Also in this case, the way to conclude the activity is defined on the basis of the type of SCO;
- adl.nav.request: allows the SCO to indicate a navigation request for processing by the LMS when the SCO is terminated. In the adaptive content package this element has been used to allow the Exit button to terminate the activity and contemporarily store the learner’s progress in the content package.

B. Extension of API functions

As pointed out in the previous sections, in order to allow the LMS to interact efficaciously with the adaptive content package, new API functions have been defined. The strengths of this work are that the API functions affect only the client side of the package, leaving the standard and the interoperability with the LMS unaltered. The defined API functions exploit the native functions of the standard to share data with the LMS.

The defined functions are:

- ris_ga(): a function that calculates the result of the first part of the ICSQ questionnaire in the cognitive style SCO. It evaluates the score for the first 9 questions, aimed at defining if the user’s cognitive style is wholist or analytic.
- ris_vv: is the twin function of the ris_ga, which aims at calculating the score for the second part of the questionnaire aimed at defining if the user’s cognitive style is verbalizer or imager. The results of both functions are stored as global variables that are used for the cognitive_style() function in order to update the user profile.
- cognitive_style(): builds the user profile that contains the ordered list of the learner’s preferred cognitive styles. According to the first cognitive style stored in the user profile, the value of cmi.score.scaled in the RTE is set. The cmi.score.scaled value is the measure threshold that allows a SCO to be activated. The values are chosen in these ranges: [0, 0.25] for the analytic style, [0.25, 0.50] for the global, [0.50, 0.75] for the verbalizer and [0.75, 1] the imager. These ranges represent a way to encode the cognitive styles in the content package. An in-depth study was made in order to assign this value to each user’s cognitive style and thus promote an efficient adaptation of the content package. In other words, the cmi.score.scaled value is updated according to the result of the CT in order to allow the content package to use the same cognitive style (if the CT is passed) or a different one (if the CT is failed). Moreover, it stores the user’s profile in a cookie so as to allow the course to be suspended and restarted at any time.

- TestSco(): calculates the result of the CT and updates the user’s profile accordingly. In other words, using the function Style_reminder() and on the basis of the user profile, if the student pass the CT the next didactic unit is shown using the same cognitive style, otherwise the reinforcement content will be supplied. In order to distinguish if a content SCO or a reinforcement SCO should be supplied, the score.scaled value is used. If it is positive the content will be supplied as discussed in cognitive_style(), otherwise a negative value is assigned according to the above described ranges: [-1, -0.75] analytic style, [-0.75, -0.50] global style, [-0.50, -0.25] verbalizer style and [-0.25, 0] for the imager.
- Style_reminder(): updates the cookies and the user profile according to the user interactions.
- The Cookie Functions are a set of functions used to write, read and to verify if the browser allows the cookies to be saved.

V. THE AUTHORING ENVIRONMENT

In order to facilitate the building of an adaptive content package our research has also defined some interventions that
could be made in one of the authoring environments used for building SCORM packages: RELOAD.

The RELOAD Project [12] is funded under the JISC (Joint Information Systems Committee) Exchange for Learning Programme (X4L) [16]. The project focuses on the development of tools that are based on emerging learning technology interoperability specifications. The main goals of the project are to facilitate the building, diffusion and reuse of LOs. Thus, the project offers a suite of software all of which is SCORM compliant.

The starting idea in our research was to propose the introduction of some facilitations into the RELOAD authoring environment that could support non expert SCORM users in building an adaptive content package. First of all, in order to modify the Organization of the package an option “Add adaptivity” could be added to its shortcut menu, which will be able to add XML code into the Manifest of the package as the child of organization.

```xml
<addAdaptivityChoice>
<addAdaptivityChoice ID="common_eq">
<addAdaptivityChoice rollupObjectivesSearch="true">
<addAdaptivityChoice rollupProgressCompletion="true" objectiveMeasureWeight="1" />
<addAdaptivityChoice rollupProgressCompletion="true" completionByContent="false" objectiveMeasureWeight="1" />
</addAdaptivityChoice>
</addAdaptivityChoice>
</addAdaptivityChoice>
```

Moreover, the following XML code will also be generated and stored in the Sequencing Collection.

```xml
<addAdaptivityChoice>
<addAdaptivityChoice ID="common_eq">
<addAdaptivityChoice rollupObjectivesSearch="true">
<addAdaptivityChoice rollupProgressCompletion="true" objectiveMeasureWeight="1" />
<addAdaptivityChoice rollupProgressCompletion="true" completionByContent="false" objectiveMeasureWeight="1" />
</addAdaptivityChoice>
</addAdaptivityChoice>
</addAdaptivityChoice>
```

After this procedure, the user will add the adaptivity to all the items within the content package. Also in this case, a new option “Add adaptivity” will be added to the shortcut of the item. At this point, the user should choose the type of item that should be added to the content package (Figure 3): cognitive style test, content SCO, reinforcement SCO or Comprehension Test.

![Figure 3. Choosing the type of SCO to be added to the adaptive content package (the labels of the buttons are cognitive style test, content SCO, reinforcement SCO, Comprehension Test)](image)

**A. Cognitive style test**

If the user wishes to add an item containing the test to define the user’s cognitive style in the tag `<item>` the following XML code will be generated.

```xml
<addAdaptivityChoice>
<addAdaptivityChoice ID="common_eq">
<addAdaptivityChoice rollupObjectivesSearch="true">
<addAdaptivityChoice rollupProgressCompletion="true" objectiveMeasureWeight="1" />
<addAdaptivityChoice rollupProgressCompletion="true" completionByContent="false" objectiveMeasureWeight="1" />
</addAdaptivityChoice>
</addAdaptivityChoice>
</addAdaptivityChoice>
```

**B. Content SCO**

If the user wishes to build a SCO content, it will be necessary to define the presentation mode used to build the content. In Figure 4 the user should choose among the different styles Analytic, Wholist, Verbalizer and Imagery according to the results of the ICSQ questionnaire.

![Figure 4. Selection of the cognitive style for the content and reinforcement SCOs (the option button labels are the different cognitive styles: Analytic, Wholist, Verbalizer and Imagery)](image)

The XML code corresponding to the Content SCO will be the following.

```xml
<addAdaptivityChoice>
<addAdaptivityChoice ID="common_eq">
<addAdaptivityChoice rollupObjectivesSearch="true">
<addAdaptivityChoice rollupProgressCompletion="true" objectiveMeasureWeight="1" />
<addAdaptivityChoice rollupProgressCompletion="true" completionByContent="false" objectiveMeasureWeight="1" />
</addAdaptivityChoice>
</addAdaptivityChoice>
</addAdaptivityChoice>
```

After the selection of the cognitive style for the content (in the example analytic) the code inserted in the Manifest will be the following:

```xml
<addAdaptivityChoice>
<addAdaptivityChoice ID="common_eq">
<addAdaptivityChoice rollupObjectivesSearch="true">
<addAdaptivityChoice rollupProgressCompletion="true" objectiveMeasureWeight="1" />
<addAdaptivityChoice rollupProgressCompletion="true" completionByContent="false" objectiveMeasureWeight="1" />
</addAdaptivityChoice>
</addAdaptivityChoice>
</addAdaptivityChoice>
```
C. Reinforcement SCO

The code inserted is similar to that described for the content SCO. The main difference is in the name of primaryObjective and in the values and ranges used for the Pre-conditions rules that are defined according to the description in section IV.

D. Comprehension Test

In this case, the reference number to the didactic unit related to the comprehension test has to be added in the Manifest. This will allow the LMS to create a link between the content or the reinforcement SCO and the CT related to them.

The code in the Manifest will be the following:

<primaryObjective name="comprehensionTest" />
<class:ruleCondition>
  <class:ruleCondition classCondition="subject/subject1" />
  <class:ruleCondition classCondition="subject/subject2" />
  <class:ruleCondition classCondition="subject/subject3" />
</class:ruleCondition>
<class:ruleCondition conditionCombination="all" />
</primaryObjective>

And this will be the code serving to define the relationship between the content and the test.

These functionalities have been designed, implemented and integrated in the RELOAD environment.

VI. CONCLUSIONS AND FUTURE WORKS

The paper describes a method for building an adaptive content package using the SCORM standard. The learning content will be adapted during the interaction according to the user’s preferred cognitive style. The basic idea underlying the defined adaptive model is that an individual can code, organize, and run content using different cognitive styles. During the learning process, on account of the domain, the motivation, etc., the preferred cognitive style can change. The theoretical model was then applied to learning content using the SCORM standard in order to assure the interoperability of the LOs. The main novelty of this work is that the SCORM specifications were not modified at all, but their primitives were used in order to implement our adaptation model. Additionally, some API functions were defined to allow the content package and LMS to apply the adaptivity on the client side of both. Moreover, a set of interventions has been built to allow the adaptive content package to be built, even by a non-expert user, using one of the most popular authoring tools, RELOAD. The next step of our research will be to set up experimentation of both the adaptation model and the functionalities in the RELOAD environment.

ACKNOWLEDGMENT

This work was supported by Regione Puglia under grant DIPIS and by the University of Bari under grant ex 60%.

REFERENCES

[16] http://www.x4l.org