Chapter 15

ITERATIVE DESIGN OF LEARNING PROCESSES

Telmo Zarraonandia, Juan Manuel Dodero, Camino Fernández, Ignacio Aedo y Paloma Díaz
Universidad Carlos III de Madrid, Departamento de Informática, Escuela Politécnica Superior

Abstract: The aim of this work is to bring together the traditional way of teaching and working using a computer-supported environment. This means, increasing the flexibility of the learning processes application, giving instructors the chance to introduce variations on runtime. Besides, learning processes are refined through its use, by making permanent the modifications which have shown to improve the learners' performance on the different learning objectives. This approach is similar to the one followed for the development of user interfaces, where the interface design is obtained by an iterative process of prototyping, testing, analyzing and refining. This chapter describes the lifecycle of the iterative design of learning processes and proposes an architecture for implementing its runtime stages for processes described by means of the IMS Learning Design specification.

Key words: Learning Design, adaptation, runtime, iterative.

1. INTRODUCTION

When describing an educational process it is not always possible to know all its elements properties at design time. Many of them as, for instance, the ones related to synchronization and temporization of the activities cannot always be established before the proper execution of the learning process begins.

On the other hand, regardless of how careful and precisely a learning process has been defined, its application to actual educational settings is all but rigid, since it is very difficult to foresee all the potential reactions from

B. Fernández-Manjón et al. (eds.), Computers and Education: E-learning, From Theory to Practice, 163–177.

learners. In practice, teachers take the learning process as a starting base, not to be followed blindly. They observe the evolution of the learners during its execution, introduce the appropriate adaptations afterwards in order to solve specific problems, reinforce the learning of some particular concepts and, more generally, guarantee the achievement of the original learning objectives. Furthermore, the adaptations proven to improve the original process results will be part of future applications. Due to the above, the learning process is traditionally refined through its use.

This work aims at increasing the degree of freedom of the teachers when applying a learning process on a computer-supported environment, offering the instructors the possibility to introduce modifications in the learning process definition during its proper execution. Those adaptive actions introduced could be evaluated against their original goal, measuring its influence on the learning objective achievement and, accordingly, giving the teacher a chance to automatically include them in the original process. This way, instructors would imitate the way teachers work in real life: the gain obtained by the use of the process is kept within the process and, at the same time, is also used to refine it.

The rest of the paper is organized as follows. First, the iterative design of learning process lifecycle will be defined, describing the purpose and characteristics of each of its different stages. Next, notations for the specification of the process evaluations and adaptations will be provided. Following, the architecture of a system able to implement the runtime phases of the iterative composition of an IMS Learning Design (IMS Global Learning Consortium, 2003) specified process will be outlined. The paper will conclude with an example of the whole late modeling process and the presentation of some conclusions and future work lines.

2. ITERATIVE DESIGN OF LEARNING PROCESS

The application of a learning process is in practice quite flexible as it is not possible to foresee all the potential reactions from the learners. Instructors take the learning process as a basis, and after observing the learner reactions, they may response providing extra examples, explanations to reinforce particular concepts, repeating activities, tuning the time-limits for completion of the assessments, etc. However, the more the instructors play the course, the less adaptation are required to be applied as the process is refined through its use. The experience gained from prior plays is comprised within the process definition and a wider range of learner reaction response is captured. This means that the course model definition does not

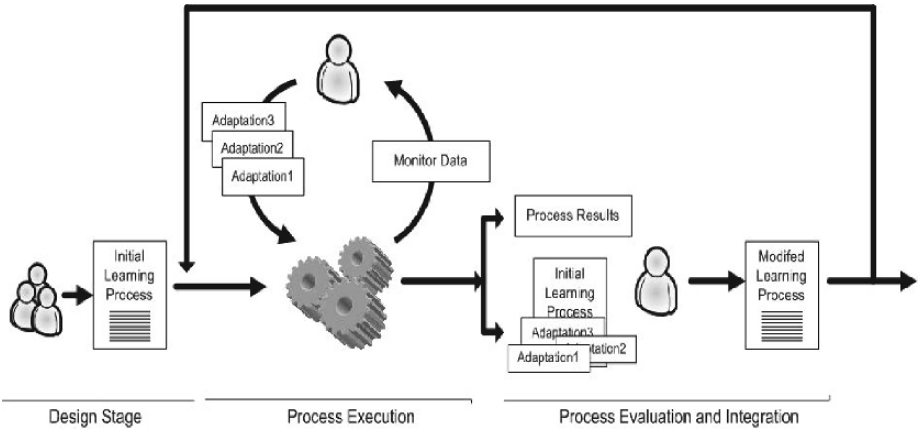

Figure 15-1. Phases of the iterative design of a Unit of Learning

conclude until no more modifications are required to be applied. This approach is similar to the one followed for the development of user interfaces, where the interface design is obtained by an iterative process of prototyping, testing, analyzing and refining (Gould et al., 1991).

Figure 15-1 illustrates the different activities of the iterative design of a learning process carried out on a computer-supported environment. The process starts once an initial model of the course has been defined and its execution begins. Instructors observe learners interactions and introduce the appropriately tagged adaptations. The success of the applied adaptations will be evaluated, and once the process is finished, the learning objective achievement will be measured. Based on that information, a new version of the learning process will be generated, including the successful modifications introduced. This new version will go through the same cycle on its next plays until no more adaptations are required to be applied.

This section provides a description for each of the different activities that compose an iterative design process: monitor the execution, adaptations introduction, adaptations evaluation, process evaluation, and finally, adaptation integration.

2.1 Monitor the execution

In order to detect potential problems and introduce the appropriate adaptive actions, it is fundamental for the instructors to be able to monitor the learner's interactions and progress during the learning process.

The more information instructors can obtain from the process execution, the better they will identify causes of problems during the learning process. For instance, if they can only retrieve information about the learner's score on

the different activities, they may only be able to conclude that her/his performance is not being adequate. Otherwise, if they could retrieve information about which resources the learner has visited and how much time she/he has spent on each of them, they may be able to extract more accurate conclusions and produce appropriate recommendations and adaptations.

On the other hand, the comparison of information from the different learning process instances of the different participants facilitates the identification of the nature of the problem.

2.2 Introduction of Adaptations

Based on the information retrieved from the monitoring activities, instructors will describe the process variations required to guarantee the process success.

Jacobson et al (1997) defined variation point as "places in the design or implementation that identify locations at which variation can occur". Variation points can be bound to the system at different stages of the product lifecycle. Svahnberg (2002) presented a taxonomy of variability realization techniques which defined different ways in which a variation point can be implemented. One of these techniques is the code fragment superimposition, where a software solution is developed to solve the generic problem; code fragments are superimposed on top of this software solution to solve specific concerns. This superimposition can be achieved by means of different techniques; as for example the Aspect Oriented Approach (Kiczales et al. 1997), and provides the designer with the possibility to bind the modifications during the compilation phase or even at runtime.

We can take these concepts into the adaptation of learning process area. The authors can describe the desired adaptations on auxiliary specification files that could be processed together with the original Unit of Learning (UoL) (IMS Global Learning Consortium, 2003) and applied at runtime giving the user the feeling that they were included in the original UoL. This way, we can maintain a single UoL definition and a number of descriptions for adaptations. Those files tie together all the changes involved in a particular adaptation and keep that particular concern separated from the main UoL functionality and the rest of adaptations.

An overview of the process is shown in figure 15-2. From several possible adaptations defined for a particular UoL, the designer chooses the one which best fits the current situation and applies it to the UoL. The introduction of the adaptive action can be carried out at design time (adaptation 1) or at runtime (adaptation 2, 3, 4). In the last case, adaptation could be applied to all the running instances of a UoL (adaptation 2), to all the users of a particular running instance (adaptation 3) or only to the personalized view of a particular user (adaptation 4).

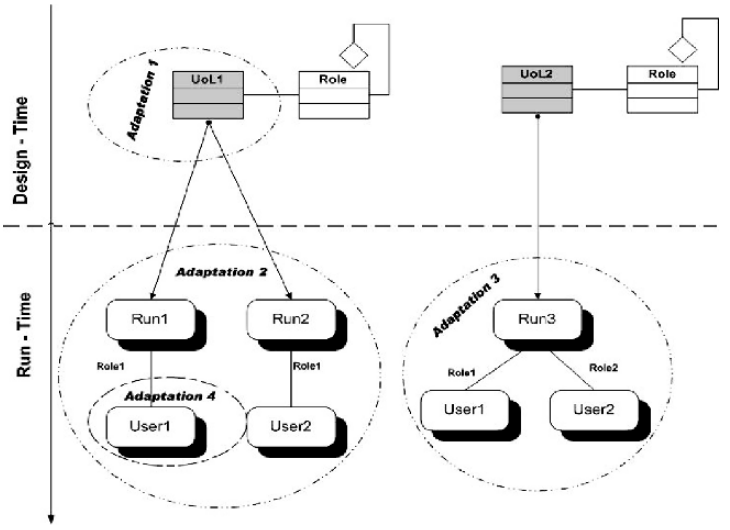

Figure 15-2. Different moments of adaptations introduction on UoL instances

We define *adaptation poke*s the description of a small modification of some elements in a learning process. A notation for the *adaptation pokes* description is provided in section 4.

2.3 Evaluation of Adaptations

To measure the success of the adaptive action it is not only necessary to evaluate the grade of satisfaction of the adaptation objectives but also check possible interactions with other parts of the course and the introduction of collateral effects. Note that the evaluation is not directly based on the learners results but on comparing the expected consequences of the adaptation with the actual ones. Hence, the difficulty lies on the identification of what is a real consequence of the adaptation and what is not. Correlation between adaptations effects must also be considered at this stage.

2.4 Evaluation of the Process

Once the learning process is finished, its results must be evaluated to identify its strengths and weaknesses. This evaluation is mostly based on the information about the performance of the learners for each learning objective obtained once the process is finished. If most of the learners score low for a particular learning objective, designers may consider including complementary material, reviewing the pedagogical approach or reviewing the calibration of the difficulty of the assessment activities. However, causes

of low performance may also lay on external circumstances or incorrect learner profiles. It is necessary then, to establish the grade of reliability of the process results by comparing them with other plays of the course data.

2.5 Integration of Adaptations

Once the process results have been analyzed, the integration phase takes place. This way, adaptive actions which have proved to mean an improvement of the process become a permanent part of it. This is a two step process: fist, instructors thoroughly examine all the adaptation results and select the ones to be integrated, and second, the system applies them to the original process design following their introductory order. Each of the adaptation introductions is validated separately. This method facilitates the identification of dependencies with rejected adaptations in case of failure.

3. LEARNING PROCESS EVALUATIONS SPECIFICATION

In order to evaluate learners' progress and process success it is necessary to explicitly specify what is going to be evaluated and how that evaluation should be performed. This specification can be provided using an XML notation for the purpose of automating its processing by the appropriate engine. Following this approach the authors present an XML schema for the learning process evaluations definition, whose graphical representation is shown in Fig 15-3. The evaluations are the core of the schema, each of them will be composed of a combination of values related to performances on process elements and learning objectives, plus another information data. An evaluation element must be provided for each learning objective. Optionally relations between the learning objectives and the process elements which contribute to their achievement can also be provided.

Three different types of elements may be required to fully specify a learning process evaluation:

- Process components: They represent elements of the learning process that contribute to a learning objects acquisition. Their definition will be composed of an identification, a reference to the corresponding learning process element and an expression to be used to estimate the learner performance for that element. This expression can be either a monitor command, for obtaining a learners' test score, for instance, or an expression where references to other components and information elements are combined to produce a value.

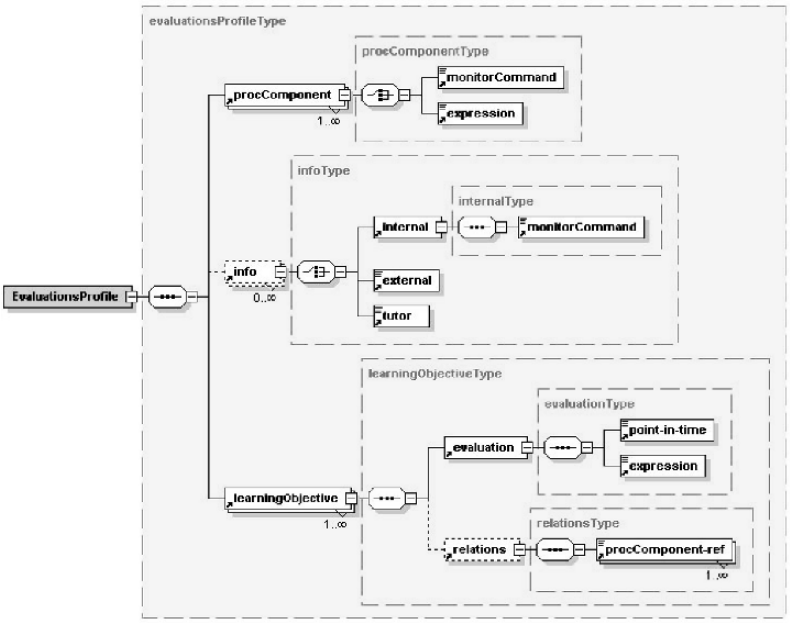

Figure 15-3. Evaluations Schema

- Information elements: Definitions of information not related with learners performance on a particular element but required for the learning objectives evaluations. This information can be obtained by monitoring the learning process, read from an external resource or introduced by the process instructor directly. This way, for instance, the number of learner's messages in the learning process's forums can be used to estimate her/his grade of interaction with other learners; results on previous learning process stored in the learner profile can by used to rate her/his improvement in a particular subject, or the instructor can be inquired regarding her/his opinion about the learner's collaborative skills.
- Learning objectives: Each of the learning objectives will be related with an evaluation and a list of process components that contributes to the learning objective achievement. The evaluation will contain the specification of the moment in time in which it should be performed and an expression which combines references to components, information elements and learning objectives with mathematical and logic operators. Once the time limit specified by the *moment-in-time* element is reached, the system engine will parse the formula, retrieve the actual value for each of the referred elements and produce the evaluation's score.

4. PROCESS ADAPTATIONS SPECIFICATION

In previous work authors (2006) defined adaptation pokes as descriptions of small modifications of some elements in a learning design process. The set of elements whose modification could be subject of description by an adaptation poke were also defined. Authors also introduced the three different types of files which could be required to fully specify an adaptation poke: an adaptation command file - describing the adaptive actions-, adaptation manifests files - containing the definition of new learning process elements-, and resource files -corresponding to new content files-.

Alternatively, the adaptation command file can be described by means of an XML notation and increased with new elements for supporting the evaluation of the adaptations. Fig 15-4 shows a graphical representation of an XML schema developed for this purposes. The schema defines three different types of elements which can be provided for the description of an adaptive poke:

- Adaptation action: It is the only mandatory element of the file. It describes the adaptive actions to be performed by the engine in charge of the adaptation interpretation. There are only three possible adaptive actions:
  - Set a value of a learning process element's property.
  - Add a new element to the structure: In this case it may be necessary to indicate the new element's parent in the structure and to provide the corresponding adaptation manifest file including its definition.
  - Remove an element from the structure.

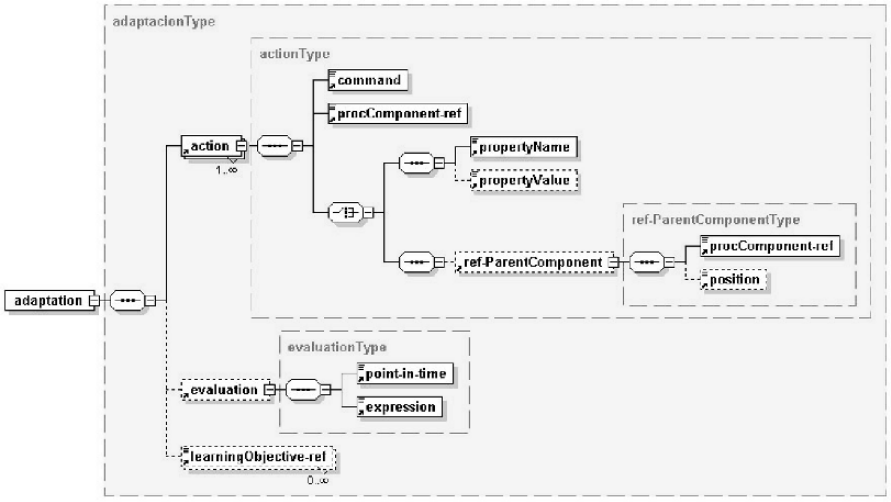

Figure 15-4. Adaptations Schema

- Adaptation evaluation: Contains a formula, similar to the ones used for the objective evaluations, which will be used to estimate the adaptation success.
- Learning objectives: As the introduction of an adaptation may influence the learner's performance of some of the learning objectives, it is convenient to specify them in this section if they are known. This will help to the adaptation influence analysis when examining the process results.

The definition of the adaptation evaluation may require the specification of new elements of the evaluations profile as process components, information elements or even learning objectives. Besides, as a result of the adaptive action, elements of the learning process may be removed and new ones added. Therefore, some of the objective evaluations may require to be updated. In these cases, it will be necessary to introduce a new type of adaptation poke file together with the above mentioned three. This file will follow the same schema as the evaluations profile definition and, as well as the specification of new elements, it can also include updates in the existing one's definitions.

5. IMPLEMENTATION OF ITERATIVE DESIGN OF IMS LEARNING DESIGN PROCESS

This section covers the proposed architecture for implementing the runtime phases for the iterative composition of learning design specified process. The core of the architecture is a Learning Design Player able to interpret *adaptation pokes* descriptions and to introduce the specified modifications at runtime. A mechanism to guarantee the integrity of the modified UoLs must also be defined.

5.1 LD Player

An Learning Design Player (LD Player) is the program that interprets a UoL. It presents the different activities and resources to the involved roles and controls their interactions. In a previous work authors (2006) outlined the structure of a LD Player capable of combining, both prior and during the execution time, the original UoL information with adaptations' descriptions included in the adaptation pokes. The proposed structure (Fig. 15-5) followed an Object-Oriented design, establishing a correspondence between the elements of the Learning Design specification and the class concept from an OO approach. It also made use of design patterns and an Aspect Oriented Approach (Kiczales et al., 1997). This allows a separate specification of the elements of the structure and the definition of the operations that can be

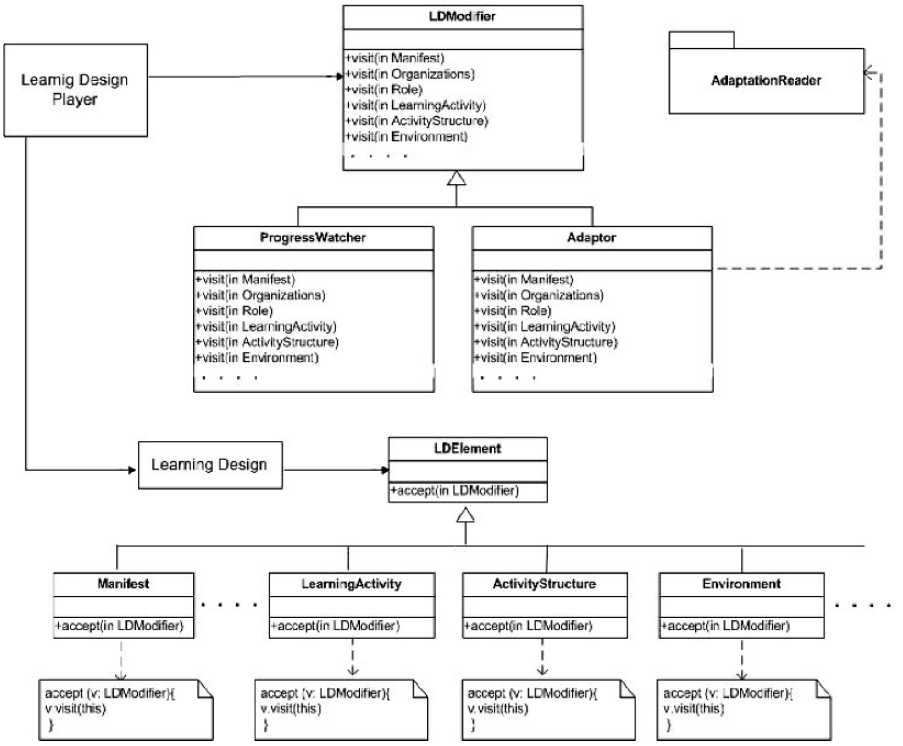

Figure 15-5. LD Player Structure

applied. Two of the possible operations that could be implemented were the modification of the elements definition (*Adaptor class*) and the retrieval of information about their stage (*ProgressWatcher*).

The *adaptation pokes* could be included in the content package or uploaded to a running instance indicating the user or UoL instances which should be adapted. The *AdaptationReader* generates the appropriate *Adaptor* object and passes it to the execution engine to perform the required adaptations.

Following the same approach, a set of commands were defined to specify the elements' characteristics whose value could be retrieved at runtime. The designer introduced the appropriate command at runtime indicating the element identifier and the UoL running instance she/he desired to observe. A *ProgressWatcher* instance was then generated and the appropriate values obtained.

The adaptive LD Player was implemented as an extension to the CopperCore IMS Learning Design engine (CopperCore, 2005) and can be used to implement the main stages of the iterative composition of learning design specified process: to introduce adaptations and to monitor the

execution. The former clearly match the adaptive LD Player operational way and the later can be performed using monitor service implementations (IMS Global Learning Consortium, 2003), complemented with *ProgressWatcher* actions. Besides *ProgressWatcher* actions can be used for the definition of the process components and information elements of the evaluations profile. The engine will generate the corresponding *ProgressWatcher* instances and the values will be retrieved when the time for the evaluation is reached.

5.2 Adaptation Validation

Some considerations must be taken into account to ensure that the adaptation by the LD Player previously described does not compromise the integrity of the original UoL. Every time a UoL is published in a particular player, a validation process is launched to guarantee its compliance with the IMS LD language definition and the availability of the referenced resources. Consequently, after the introduction of runtime adaptations, the same validation process should be repeated to ensure that the UoL definition remains valid.

Lama (2005) and Amorin (2006) described an IMS LD-based ontology which captures the semantics of the IMS LD specification as well as the restrictions to be verified between the LD concepts. As this ontology model defines formally these restrictions, it is possible to use it for the detection of inconsistencies on adapted instances of a learning design, because the detection of inconsistencies will happen when these restrictions are not verified. The IMS LD ontology was implemented in Frame-based Logic (F-Logic) (Kiefer et al., 1995), and the FLORA-2 reasoner (Yang et al., 2005) was used to check the axioms of the ontology when the concepts instances were introduced.

The process for detecting the inconsistencies can be resumed as follows: first, an *adaptation poke* is introduced into UoL instance, and, as a consequence, its LD description is changed; then, the ad-hoc translator is executed to transform the XML-schema representation of the adapted learning design into the F-Logic description; and finally, the FLORA-2 reasoner is invoked to answer the queries associated to the axioms that must be verified.

5.3 Evaluation and Adaptation Example

In order to make a clear understanding of the above described process, authors illustrate some adaptations and evaluations of an example course.

Consider a unit of learning which covers different topics on the *Data Structure & Algorithms* subject (Fig 15-6). The course is composed by theoretical activities and autoevaluation exercises. On an scheduled date tests and problems become available for the learners final evaluation.

```
<EvaluationsProfile>

 <internal id="i-AutoTest" monitorCommand="getPropertyValue Auto.SCORE"></internal>
 <internal id="i-InsDel" monitorCommand="getPropertyValue Final.SCORE"></internal>
 <tutor id="i-Exercise" Text="Introduce the score for the exercise:"></tutor>

 <procComponent id="e-Definition" design="LD-Definition" monitorCommand="getPropertyValue Basics.SCORE"/>
 <procComponent id="e-Insertions" design="LD-Insertion" expresion="i-InsDel / 2"/>
 <procComponent id="e-Deletion" design="LD-Deletion" expresion="i-InsDel / 2" />
 <procComponent id="e-Rotation" design="LD-Rotation" datatype="integer"
                                                  monitorCommand="getPropertyValue Rotations.SCORE"/>
 <learningObjective id="lo-Theory" description="AVL trees theory understanding">
    <evaluation>
       <point-in-time command="LD-Final on-completion"/>
       <expression expresion="e-Definition*0.2+i-InsDelTest*0.4+e-Rotations*0.4" />
    </evaluation>
    <relations>
       <procComponent-ref ref="e-Definition"/>
       <procComponent-ref ref="e-Insertion"/>
       <procComponent-ref ref="e-Rotations"/>
       <procComponent-ref ref="e-Deletions"/>
    </relations>
 </learningObjective>

 <learningObjective id="lo-Practice" description="AVL trees practical application">
    <evaluation>
       <point-in-time command="LD-Final on-completion"/>
       <expression expresion="i-Exercise" />
    </evaluation>
    <relations>
       <procComponent-ref ref="e-Insertion"/>
       <procComponent-ref ref="e-Rotations"/>
       <procComponent-ref ref="e-Deletions"/>
    </relations>
 </learningObjective>

</EvaluationsProfile>
```

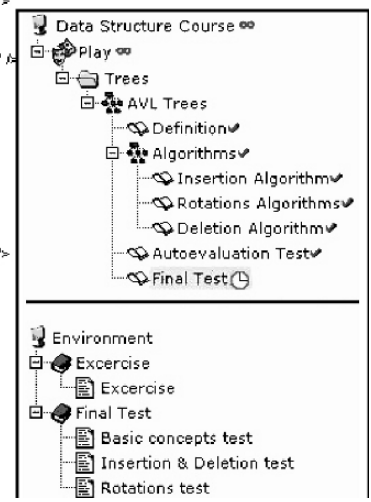

Figure 15-6. Evaluations Profile and Course Structure

The evaluation profile shown in Fig 6 has been defined for the course. The process components correspond to the activities that cover the theoretical aspects of the course. The information elements in turn, retrieve information about the learners test and exercise scores by using ProgressWatcher commands or directly inquire the tutor. Finally, two learning objectives are defined: theoretical understanding and practical application. Their estimation is obtained using the previously described elements and should be calculated after the completion of the latest activity of the course.

During the course execution two adaptive pokes (Fig. 15-7) are required to be executed. The first one is included as a result of the poor performances in the autoevaluation test. It introduces a new environment including a set of applets containing visual animations of the presented algorithms. The adaptation goal is to improve learners understanding of the subject. To measure this goal achievement the results on the final exercise and tests will be used. The adaptation introduction should be considered when analyzing the two learning objectives of the course.

```
<adaptation>

 <action>
  <command expression="add"/>
  <procComponent-ref ref="LD-Animation"/>
  <refPadre>
    <element-ref ref="LD-Algorithms"/>
  </refPadre>
</action>

<evaluation>
   <point-in-time command="LD-Final on-completion"/>
   <expresion expresion="e-Definition*0.1+i-InsDelTest*0.2
                        + e-Rotations*0.2+i-Exercise*0.5"/>
</evaluation>

<learningobjective-ref ref="lo-Theory"/>
<learningobjective-ref ref="lo-Practice"/>

</adaptation>

</xs:schema>
```

```
<adaptation>

<action>
   <command expression="set"/>
   <procComponent-ref ref="LD-Final"/>
   <property-name ref="completion"/>
   <property-value ref="false"/>
</action>
<action>
   <command expression="set"/>
   <procComponent-ref ref="LD-Final"/>
   <property-name ref="time-limit"/>
   <property-value ref="PT45M"/>
</action>

<evaluation>
    <point-in-time command="LD-Final on-completion"/>
    <mathExpresion expresion="e-Rotations not null"/>
</evaluation>

</adaptation>
</xs:schema>
```

Figure 15-7. Adaptation Pokes Examples

The second adaptive poke is introduced after detecting that most learners have not been able to complete the evaluation on the scheduled time. Therefore, the estimated time for the activity completion is not accurate and should be adjusted. The adaptation will be considered as successful if values for the latest test are obtained, meaning learners finished the task. The adaptation does not contribute to the process learning objectives achievement as such.

Once the process is finished the integration process takes place:

- Learners' results are evaluated: scores on both learning objectives are over 7 out of 10 for the 70% of them. Instructors consider the results as satisfactory.
- The two adaptations are also evaluated:
  - The fist one was marked as related with both learning objectives. The adaptation evaluation is satisfactory and therefore instructors label it to be integrated.
  - Values are obtained for the latest test from most learners. Therefore the evaluation of the second adaptation results *true* for most of them. Accordingly instructors label this adaptation for the integration process.

6. CONCLUSIONS AND FUTURE WORK

This paper has introduced the concept of *adaptation poke* as the specification of small adaptive actions that can be applied, even at runtime, to a previously defined learning process. The adaptive actions introduced are evaluated against their original goal, measuring its influence and, consequently, giving the instructor a chance to automatically include them in the original process. The cyclic process of refine the learning process definition by its use is called iterative design.

An architecture of a Learning Design Player that provides the means to implement the runtime stages of the iterative composition in IMS Learning Design specified process has been described. The player was designed as an extension to the CopperCore runtime engine and implemented with the help of different design patterns and an Aspect Oriented Programming approach. Once the UoL has been adapted, it must be validated in order to guarantee its compliance with the IMS LD specification. For that purpose an ontology that captures the semantics of the elements of the Learning Design specification is used. To help on the evaluation of both the introduced adaptations and the process results, an evaluation model which works on top of the learning process definition can be used. A notation for this evaluations specification has also been provided.

An application to aid on the iterative design process is currently being developed. On one hand, the application will facilitate the authoring of process evaluation profiles and adaptations. By using a GUI interface, designers will be able to select elements of an IMS LD specified process and connect them with evaluation profile elements. Templates to facilitate the adaptation definitions and new learning process components specification will also be provided. On another hand, the application will communicate with a CopperCore engine increased with adaptation capabilities in order to directly introduce the described adaptations into running UoL instances. The retrieval of data to populate the evaluation profiles will also be possible by means of a *ProgressWatcher* implementation. This will simplify the process progress monitorization tasks.

ACKNOWLEDMENTS

This work is part of the MD2 project (TIC2003-03654), funded by the Ministry of Science and Technology, Spain.

REFERENCES

IMS Global Learning Consortium, 2003, IMS Learning Design information model, version 1.0-final specification; http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html

Gould, J. D., Boies, S. J. and Lewis, C., 1991, Making usable, useful, productivity: enhancing computer applications *Communications of the ACM*. **34**(1), pp. 72-85.

Jacobson, I., Griss, M. and Johnson, P., 1997, *Software Reuse. Architecture, Process and Organization for Bussiness Success* (Addison Wesley).

Svahnberg, M., Gurp J. van and Bosch. J., 2002, A taxonomy of variability realization techniques, Blekinge Institute of Technology, Technical paper, Sweden

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., and Irwin, J., 1997, Aspect-Oriented Programming, *Proceedings of the European Conference on Object-Oriented Programming,* 1241, pp. 220–242

Zarraonandia, T., Dodero, J. M. and Fernández, C., 2006, Crosscutting runtime adaptations of LD execution, *Journal of Educational Technology and Society*, **9** (1), pp. 123-137

CopperCore, 2005, CopperCore v2.2.2 (OUNL) release; http://coppercore.org/

Lama, M., Sánchez, E., Amorim, R. and Vila, X.A., 2005, Semantic description of the IMS Learning Design Specification. *AIED-Workshop on Semantic Web technologies for E-Learning (SW-EL 05)*, Amsterdam, pp. 37-47.

Amorim, R., Lama, M., Sánchez, E., Riera, A. and Vila. X.A., 2006, An ontology to describe semantically the IMS Learning Design Specification. *Journal of Educational Technology and Society*, **9** (1), pp. 38-57.

Kiefer, M., Lausen, G. and Wu, J., 1995, Logical foundations of object-oriented and frame-based languages, *Journal of ACM*, **42**(4), pp. 741-843

Yang, G., Kiefer, M., Zhao, C. and Chowdhary, V., 2005, FLORA-2: users' manual; http://flora.sourceforge.net/docs/floraManual.pdf.