Generative Adaptation and Reuse of Competence Development Programmes

Juan Manuel Dodero [1], Telmo Zarraonandia [2], Camino Fernández [2], David Díez [1]

[1] Computer Science Department, University Carlos III of Madrid Av. de la Universidad Carlos III 22, 28270 Colmenarejo, Madrid Spain	[2] Computer Science Department, University Carlos III of Madrid Av. Universidad 30, 28911 Leganés, Madrid Spain

Abstract: Instructional engineering provides methods to conduct the design and adaptation of competence development programmes by the combination of diverse learning components (i.e. units of learning, learning activities, learning resources and learning services). It occurs through an established process workflow in which models with diverse levels of abstraction are used to depict such learning components. This paper presents a model-driven generative method used to adapt and reuse a set of learning components for the delivery of a competence development programme concerned with a given learning objective and which serves for a specific instructional context.

Keywords: Instructional Engineering, Adaptable Units of Learning, Generative Software Engineering.

1 Introduction

Competence Development Programmes (CDPs) are collections of learning activities and units, which are used to increase the overall effective performance of a learner within a certain task. CDPs can be facilitated by shared, self-organized networks of IMS LD-based Units of Learning (UoL), which rely on diverse web-based technologies to be realised (Herder et al., 2006). In the research of how learning networks can be specified, built, and tailored to suit an individual learner's needs, several adaptive web-based educational systems have been studied (Brusilovsky, 2003). Such systems aim at intelligently incorporating and performing certain activities traditionally executed by human teachers. Nonetheless, adaptations occurring during learning time must be designed beforehand, which involves the complexity of building and mixing models of the goals, preferences, and knowledge of the learners.

Although the design of an adaptable CDP is difficult to be completely automated, a computer-aided method can help in designing and adapting the programme. Such methods are a part of Instructional Engineering (IE) discipline (Paquette, 2004), which aims at increasing the degree of automation of the instructional design process. This paper describes a generative IE method used to create and adapt CDPs. Such adaptations can be realised in design-time (i.e. before the actual programme begins executing) or in learning-time (i.e. after the beginning), which complicates the instructional design task.

When engineering a CDP it is not always possible to know in detail its structure and all of its components beforehand. Some details cannot be completely established before the learning process begins. Regardless of how carefully a CDP has been defined, its actual application is all but rigid, since it is difficult to foresee all the potential reactions from learners. In practice, learning designers take the learning design as a starting point not to be followed blindly. They usually observe the evolution of learners, and afterwards introduce the appropriate adaptations to reinforce some aspects of the CDP, as long as the achievement of the original learning objectives is guaranteed. Most IE methods are used to define learning programmes from scratch. This paper explains a generative IE method that also facilitates adapting an existing CDP from further requirements stated by the instructional engineer.

The rest of this paper is structured as follows: the remainder of this section describes the related work and methodological issues of designing and adapting CDPs and UoLs. Then, section 2 explains a

generative IE method, which is put into the framework of current model-driven, generative engineering practices. Section 3 includes a case study on how the IE method has been applied to adapting a UoL to convert it to a simple CDP. Finally, section 4 ends up with some conclusions and future work.

1.1 Related work

Instructional Engineering is defined as a method that supports the analysis, design and delivery planning of learning systems, integrating the concepts, processes and principles of instructional design, software engineering and cognitive engineering (Paquette, 2004). The discipline of IE has its roots in Tennyson and Barro (1995), who described earlier attempts to automate instructional design. More recently, Sloep, Hummel and Manderveld (2005) have claimed learning design procedures to be similar to those of the traditional software engineering life cycle - namely analysis, design, development, implementation and evaluation. Early IE methods, such as the Courseware Engineering Methodology (CEM), the *Méthode d'Ingénierie des Systèmes de Apprentissage* (MISA) and the Instructional Software Development Process (ISDP) organize the work in iterative and incremental development cycles, as they are heavily influenced by software engineering practices. First, the CEM method (Uden, 2003) divides the courseware engineering life cycle into three processes (i.e. inception, construction and evaluation), which unfold into four main phases (i.e. analysis, design, development and evaluation). The artefacts obtained are classified into four models (i.e. pedagogical, conceptual, navigational and interface model). Second, the MISA method (Paquette, Aubin and Crevier, 1999) manages the production of a learning system through six phases (i.e. definition, preliminary analysis, architecture, conception, realization and validation, and dissemination), which are developed along four orthogonal axes (i.e. knowledge model, pedagogical model, media model, and delivery model). Thirdly, the ISDP method (Demirors et al., 2000) is an adaptation of the ISO/IEC 12207 software life cycle process, which defines a core set of activities used to transform requirements into a consistent set of artefacts that represents an instructional product (i.e. requirements specification, design, implementation and testing, and project management). These core processes define a standard set of intermediate products to be delivered.

All the methods described above are based on the analysis of the learning system from different perspectives and levels of abstraction, which constitute a widely used technique to manage complexity. They also define a number of models to depict different aspects of the learning system design. Nevertheless, the treatment of dependencies between cross-cutting, orthogonal views (or models) of the learning system is rather limited.

1.2 Methodological issues

Designers of a CDP have the goal of defining a programme that satisfies a certain learning need and produce some learning outcomes. They provide the input to the IE process showing different *abstraction levels*, targeted on different *contexts*, and affected by different *concerns*, which are described as follows:

- An instructional designer can specify parts of the desired CDP with a different *level of abstraction* from other designers. For instance, an instructor can state the need to design a "90-minute long, 60-item, multiple-choice, multiple-answer quiz assessment" activity or she may want to design a "long and difficult assessment." Furthermore, she can select the questions in the quiz by hand, according to their difficulty level, or use a computer-assisted quiz composition tool to do that task.

- Although reusability is a very desirable goal (Polsani, 2003), instructional designers should not try to design universally valid CDPs. Instead, the programmes should be targeted to a specific *context*. A major difficulty in achieving reusability in CDPs is that the programmes are aimed for functioning in different learning contexts (Koper, 2003). For instance, the design of a course on Descriptive Statistics should not be the same for Electronic Engineering and for Social Sciences' students. After that course is designed, could it work also for Computer Science studies? Such context-related issues have to be considered.

- Rarely the learning goal of a CDP is unique, but it usually consists of a number of goals. In addition, the attainment of these goals can be required with different accuracy levels. Both the extension of learning goals (i.e. the number of goals) and their intensity (i.e. the level of attainment) make up the instructional *concern* of a designer. For instance, a project-oriented course on software engineering can be mainly concerned with learning the planning and estimation procedures, or it can be focused only on managing the development method and tools. Yet, both goals can be expected at the same time but with different levels of fulfilment for each one. Concerns are usually structured as taxonomies of learning objectives, which can yield quite different programmes.

Abstraction level issues are usually managed in common IE methods by defining a number of design models, which are targeted to different roles and development stages, as described in the previous section. Some IE approaches turn to the aspect-oriented paradigm and represent these issues as cross-cutting concerns. For instance, aspect-oriented techniques have been purported to re-engineer educational material (Pankratius and Vossen, 2005). The separation of concerns principle suggests considering multiple perspectives on the engineered material. This helps in solving dependencies and overlaps between the orthogonal models of a systematic IE method. Nonetheless, dealing with a set of design models entails further questions about how and when merging them to yield the final CDP. Furthermore, it is advisable that the resolution of such dependencies be carried out as early as possible in the method, as proposed by Díez, Fernández and Dodero (2006), who describe how model-driven and aspect-oriented analysis can be combined to define an effective learning analysis method.

To manage the context issue, learning design patterns (McAndrew, Goodyear and Dalziel, 2006) are the mainstream trend. Patterns are parameterized templates that can be adapted to the concrete learning context. Yet, some open issues here are: how to select the adequate learning design pattern(s); how to merge or combine the patterns; and if such patterns can be actually combined in the desired context (Hernández-Leo et al., 2006a).

Finally, regarding modelling the instructional concerns, scarce approaches have been provided; and most of them have been based on the pattern concept as well. Although Derntl and Botturi (2006) use a goal-based approach to structure instructional requirements, they focus on pattern identification, application and maintenance. They do not aim at facilitating the adaptation of an instructional requirement that is quite similar to desired one, but not completely satisfactory - for instance, a project-oriented course which is in a repository of readily available CDPs can be concerned with learning software engineering development methods, but not learning any tool; and learning of a given tool is now required as an additional goal in the adapted CDP.

2 Model-driven generative adaptation

Complete automation of the design and adaptation of a CDP is a difficult task. Nevertheless, instructional engineering methods can help in designing and adapting an originally available programme. This section describes a generative instructional engineering method, which considers the methodological issues described above along the engineering lifecycle. As a first step, a *generative domain model* is defined to serve as the framework of the method. Next, the method itself is explained.

2.1 Generative domain model

Design knowledge affecting a CDP can be organized into a network of related *domains* (e.g. technical, didactic, presentation, etc.) Each of these domains must be expressed in a specific, high-level domain language (e.g. web-based application languages, pedagogic design patterns, usability descriptions, etc.) and also comply with a certain *meta-model*. The CDP is defined with a set of high-level specifications describing different aspects of design. The generative IE process encourages the efficient use of system models by engineering *families* of *learning components* and *services* that eventually become part of one CDP. The members of the family are generated based on a common *generative domain model*, i.e. a model of system family consisting of three elements, namely (1) a means of specifying family

members; (2) the components from which each member can be assembled; and (3) the configuration knowledge used to generate a finished member from higher-level specifications.

Family members are concrete learning components or services. They are not templates or patterns, but final learning design solutions, which can range from ready-to-run UoLs to complete chunks of different granularity (Hernández-Leo et al., 2006b). A higher-level specification of these family members must be defined. These are provided by *feature models*, which are structured collections of eligible features, including formalized definitions of features, composition rules (i.e. indication of which feature combinations are valid, including requirements and exclusiveness); and rationales for features (i.e. reasons for choosing or not a given feature.) The annotation of features with rationales when these are applied to a certain design is essential to feed and grow up the configuration knowledge base. Moreover, feature models allow building service-level agreements on a CDP, so that instructional designers (or even learners) can adapt the required level of quality for a learning service, and this can be traced from the initial learning objectives to final assessments.

Second, components that make up a CDP consist of available learning designs, units of learning and learning services (Koper, 2003), which must be found and assembled. A way to find such components is annotating them with ontological representations of competences (Dodero, Sánchez-Alonso and Frosch-Wilke, 2007). Another way is explicitly capturing the required adaptations which occur after the deployment of a CDP, and packing them as *adaptation pokes* (Zarraonandia, Dodero and Fernandez, 2006). These can be stored separately from other components, annotated, and afterwards applied to similar learning design situations. In this paper we have used the latter approach.

The selection of features that make up a component is part of the adaptation process, which can be realised during learning time as well as in design time. For instance, the instructional engineer can decide on an initial feature model to design a CDP positioning service, whilst deferring the consideration of features that affect the CDP navigation service until the learning-time.

2.2 Model-driven development

To generate a finished member from a number of member specifications, we use a model-driven development approach. We have considered the development layers of Robson, Collier and Muramatsu. (2004) for designing reusable collections of digital resources:

- Pedagogy: how a digital learning resource is used as part of a learning strategy or instructional design. Models of this layer usually include the pedagogical method, the behavioural or constructive features of learning, the cognitive level of learning objectives (e.g. expressed according to Bloom's cognitive taxonomy) and their traceability, the effort or time required to run the programme, etc. The underpinning meta-models are usually built by ontological engineering (Mizoguchi and Bourdeau, 2000), such as, for instance, Murray (1996) and Leidig's (2001) educational ontologies.

- Structure: how a digital learning resource is structured into learning components, and how these are connected between them. In this layer, IMS or other specifications that define the structuring of contents, either static (e.g. IMS Content Packaging) or navigational (e.g. IMS Simple Sequencing or IMS LD level B features) have to be taken into account as meta-models.

- Content: the information that is contained in the learning resources and that is intended to affect a change in cognitive state. This layer considers learning resources as providers of specific knowledge on a certain subject or discipline. Although the issues to be modelled in this layer are subject-specific, reusability can be improved if meta-models are provided as a combination of upper ontologies and domain-specific ontologies (Niles and Pease, 2001).

- Presentation: how a resource is rendered and what interactive elements will be used to render it. That involves how well users respond to the user interface and usability features of the resources, specially related to time on task, accuracy, recall, and emotional response. Feature-

based meta-models have been used to capture such usability characteristics (Fey, Fajta and Boros, 2002).

- Context: cultural, academic, organizational, and other factors that are needed to properly interpret a digital learning resource. Ontologies have been also used as the meta-model to explicitly represent the actual context of use of a learning resource in a learning design (Jovanović et al., 2006).

Our model-driven development framework is focused in the first four layers. These make up the concerns of the grid depicted in Table 1, which categorizes the software development models or *artefacts*, according to Greenfield and Short (2004). Columns in the table define the development layers, while each row defines a level of abstraction. Each cell depicts different kinds of models as the potential sources for the development artefacts. The context layer has not been included in the table, since it is rather complex and should be considered separately. Although other layers could have been considered, they are general enough to include other dimensions that could have been taken into account when designing a CDP.

	Pedagogy	Structure	Content	Presentation
Higher abstraction level	Narrative Use cases	UoL structure CDP services	Knowledge Competences Attitudes	web mobile digital TV
Lower abstraction level	Didactic method Role definition	Activities Environments Role play	Related concepts and skills	Supporting technologies

Table 1. A grid for categorizing instructional engineering artefacts

The generative scheme used to transform between models in each cell is depicted in Fig. 1. A set of independent models are gradually merged, mapped and transformed in order to obtain another model. In the scheme, models *A* and *B* are conformed to respective meta-models. The output model *C* corresponds to the learning component whose generation is intended. Model *C* is conformed to an either explicit or implicit meta-model resulting from the combination of *A* and *B* meta-models. For instance, in the higher-level row, a CDP navigation service (i.e. a model conformed to a structure meta-model) and a web-based navigational model (i.e. a model conformed to a presentation meta-model) can be mapped and transformed to generate a web-based implementation of the service (i.e. a model conformed to a non-explicit meta-model resulting from combining structure and presentation ones). In further stages of the development, this web-based implementation can be transformed from the higher to a lower level of abstraction - for instance, by implementing navigation among activities with AJAX and Java *Struts*. These steps can be iteratively taken along the engineering lifecycle in order to generate a learning component that eventually complies with several meta-models.

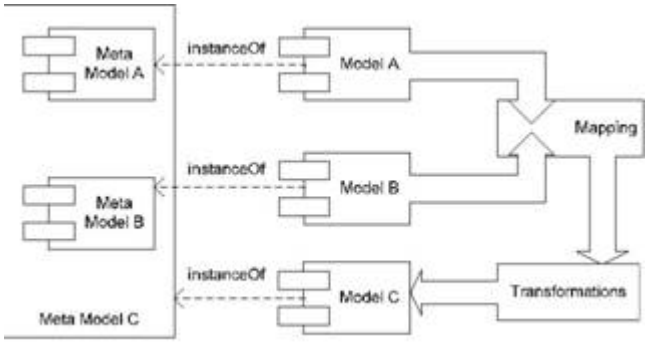

Figure 1. Model-driven generation of a software development artefact

Nevertheless, models are not completely independent and unrelated from each other. For that reason, model mappings and transformations are not straightforward. Although transformations are undertaken by metadata-based transformations, they cannot be carried out without the help of the learning designer.

Having all these rationales in mind, the generative IE method can be summarized in the following iterative phases, namely the instructional engineering workflow:

- The *analysis* phase aims at defining an anticipated view of the learning system to be built, in order to have a better understanding of the context in which it will be used and to optimise the development process. Then, an analysis model is produced.

- The *design* phase translates the delivered analysis model into a number of abstract components (i.e. learning components and services) that guarantee the service-level agreement derived from the analysis phase.

- The *implementation* phase builds the concrete learning components and services to fulfil the abstract components enclosed in the design model.

- The *evaluation* phase closes the iteration, compiles the design rationales occurred during the previous phases, and annotates the components for further improvements of the method.

Verification and validation are realized along analysis and design phases. These are not a separate phase, but they are orthogonally included into analysis and design activities. When verification and validation tasks are done during the analysis phase, they may affect either the goal selection or the configuration of features from the feature model. If goals or features selected would lead to conflicting or non-compatible configurations, feature restrictions could be applied. This can be done through basic feature modelling, which can be thought of as a feature hierarchy plus a propositional formula, as described by Czarnecki and Kim (2005). An alternative form is using rich ontology modelling, for which feature models are only views on ontologies (Czarnecki, Kim and Kalleberg, 2006).

3 Case study: generative adaptation of a unit of learning

In the following, the instructional engineering method is exemplified and applied to adapting an original CDP that was designed as a simple unit of learning, which consists of two activities, as shown in Fig. 2. In this case study, an available CDP is adapted to assist regular students in enrolling to a learning organization, as well as they acquire a set of basic knowledge competences.

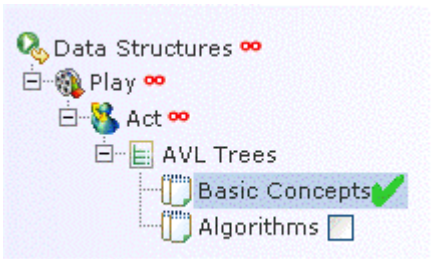

Figure 2. Original UoL consisting of two sequenced learning activities; this UoL will be adapted to build a simple CDP which includes a number of assessments and a monitoring service

Although the actual case is more detailed than what is depicted here, all the details of the model-based development process are mot shown. In summary, the following meta-models have been used: for the analysis phase, a CDP feature model is used, which covers structure and content development layers; for the learning design phase, IMS LD and an IMS LD-based ontology (Amorim et al., 2006) are used, extending across the two abstraction levels of structure and content layers; and for lower-level implementation, an specific adaptation model (Zarraonandia et al., 2006] was used to express

adaptations to the learning components selected in the design stage. For the sake of illustrating the IE method steps, in the following we will focus on describing how the original CDP is adapted through a full iteration of the workflow.

3.1 Learning analysis

First, a number of potential goals must be provided at the beginning of the analysis phase. For simplicity, we considered only three goals for a student, namely: enrolling as a newcomer, engaging in the university, and becoming an assistant student. The selection of one goal triggers the analysis. Fig. 3 depicts the feature model used for the specification of CDP family members in this phase. The CDP feature model includes mandatory features (e.g. `competenceDevelopmentProgramme`, `competence`, `assessmentService`, `navigationService`, and `learnerSupportService`); sub-features (e.g. `level`); optional features (e.g. `positioningService`); exclusive-or feature groups (e.g. `knowledgeElement`, `attitude` and `skill`) and non-exclusive (e.g. `portfolio`, `formative` and `summative`); and feature configurations. For details on feature modelling, see Czarnecki and Antkiewicz (2005). Some features derive from the main supporting component services of a CDP —namely positioning, navigation, assessment and learner support (Herder et al., 2006)—.

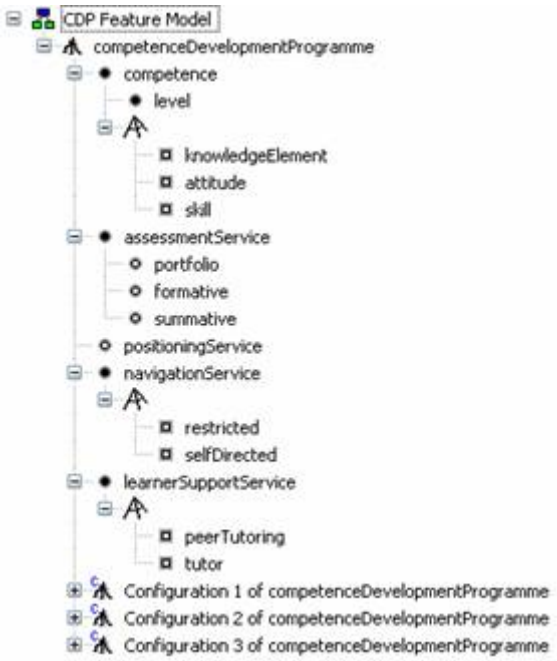

Figure 3: A feature model for the specification of CDP family members

Aftterwards, goals are translated into *feature configurations*, which are groups of features commonly selected altogether. For instance, the `enrolAsNewcomer` goal is mapped to the feature configuration number 4 (see Fig. 4), which consists of the following elements: a simple competence of type `knowledge`; both formative and summative assessments required for each learning activity; positioning service required; navigation service allowed with any option; and tutor-based learner support service. Although we have omitted details about competence level and other sub-features of `competence`, they could have been easily taken into account in the analysis as well -for instance, to determine the level of fulfilment required for the feature. A similar process can be also made on a learning activity basis to refine the number of assessments introduced; in our case, all activities are undergone to formative assessments.

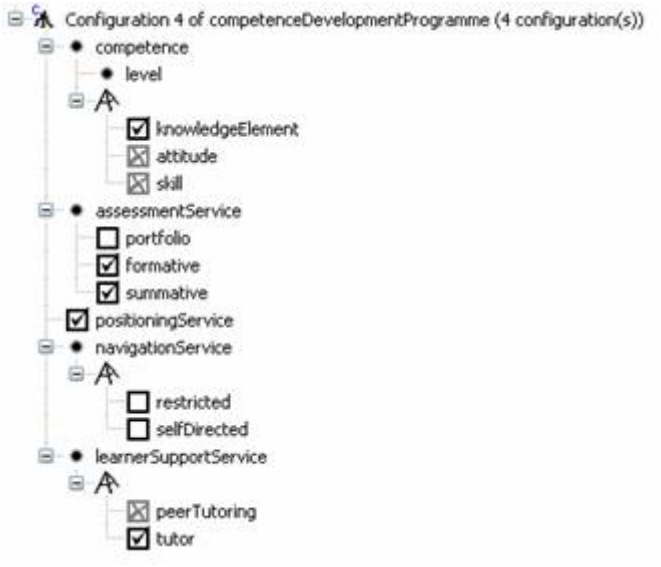

Figure 4. A configuration of CDP to achieve the goal enrolAsNewcomer

3.2 Learning design

The CDP that is being adapted should assist students to achieve the goal enrolAsNewcomer with a set of competences provided by the feature configuration selected in the analysis phase. The learning design phase involves mapping these abstract features to available units of learning, learning activities or learning services —namely, learning design components—, which are suitable to provide the required adaptation. Such learning design components are implemented as adaptation *pokes*, which are simple, run-time adaptations that can be applied to an EML-based existing unit of learning to modify or adapt some behaviours of the UoL (Zarraonandia et al., 2006). This process is supervised by the learning designer, who can observe the execution of the current version of the UoL, and define an appropriate number of adaptation pokes. In the example, adaptation pokes are combined to define more complex adaptations according to the feature configuration of Fig. 4.

The feature configuration provides two pokes that add formative and summative assessments to each activity [1]; one poke that adds a pre-test evaluation to position students in the course; and one poke that adds a monitor service to let the tutor supervise students' activities. These adaptations are described as follows:

```
<poke id='add_assess_formative'>
  <actions>
    <insertion>
        <idElement-Ref ref='formative-assessment-activity-1'/>
        <ParentElement-Ref ref='AVLTrees-Activity' pos='2' />
    </insertion>
    <insertion>
        <idElement-Ref ref='formative-assessment-activity-2'/>
        <ParentElement-Ref ref='AVLTrees-Activity' pos='last()' />
    </insertion>
  </actions>
</poke>

<poke id='add_assess_summative'>
  <actions>
    <insertion>
        <idElement-Ref ref='summative-assessment-activity-1'/>
        <ParentElement-Ref ref='AVLTrees-Activity' pos='last()'/>
```

```
    </insertion>
  </actions>
</poke>
```

```
<poke id='add_positioning_service'>
  <actions>
    <insertion>
        <idElement-Ref ref='pretest-assessment-activity-1'/>
        <ParentElement-Ref ref='AVLTrees-Activity' pos='first()'/>
    </insertion>
  </actions>
</poke>
```

The following IMS-LD example depicts the part of the UoL content before the adaptation poke was applied, while Fig. 5 shows the required structure of the activity.

```
<imsld:activity-structure identifier="AVLTrees-Activity" sort="as-is"
                                                structure-type ="sequence">
   <imsld:title>AVL Trees</imsld:title>
   <imsld:learning-activity-ref ref="Basics-Activity" />
   <imsld:learning-activity-ref ref="Algorithms-Activity" />
</imsld:activity-structure >
```

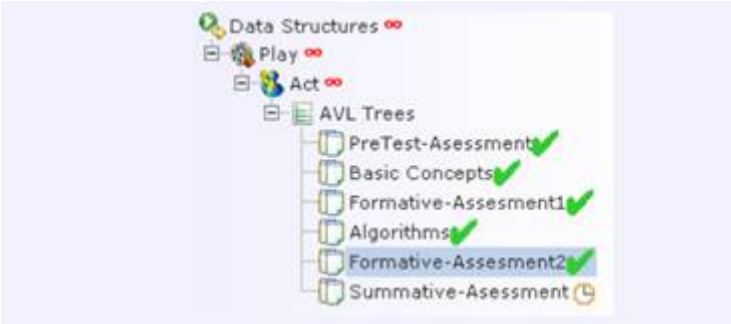

Figure 5. Definitive UoL consisting of several learning and assessments activities.

The next activity demonstrates the result of adapting the UoL to make up the CDP.

```
<imsld:activity-structure identifier="AVLTrees-Activity" sort="as-is"
                                                structure-type ="sequence">
   <imsld:title>AVL Trees</imsld:title>
   <imsld:learning-activity-ref ref="PreTest-Assesment" />
   <imsld:learning-activity-ref ref="Basics-Activity" />
   <imsld:learning-activity-ref ref="Formative-Assesment1" />
   <imsld:learning-activity-ref ref="Algorithms-Activity" />
   <imsld:learning-activity-ref ref="Formative-Assesment2" />
   <imsld:learning-activity-ref ref="Sumative-Assesment" />
</imsld:activity-structure >
```

On the other hand, the following adaptation poke is used to add the tutor-based monitoring service that was required in the analysis phase:

```
<poke id='add_tutor_monitor'>
  <actions>
    <insertion>
        <idElement-Ref ref='tutor-monitor-service-1'/>
        <ParentElement-Ref ref='//imsld:environment[@id="general"]'
```

```
            pos='last()' />
    </insertion>
  </actions>
</poke>
```

There are several possibilities of adding the tutor-based monitor service. The first possibility applies when the original UoL does only contain learning activities. Then, it is desirable to add a new tutor role and a role-part that plays a new support activity within the new environment containing the monitor service. However, if the original UoL already contains a tutor role and an adequate support activity, they can be selected and adapted by the poke (for instance, by adding a new environment to the support activity that contains the monitor service). Finally, if a tutor role, a support activity, and an adequate environment are readily available in the UoL, the adaptation can be only adding the monitor service to the environment.

3.3 Verification and validation

Verification and validation activities are orthogonally included into analysis and design phases. In our case study, basic feature modelling techniques have been applied during analysis, as well as ontology modelling has been carried out during design. During the design phase, an IMS-LD ontology (Amorim et al., 2006) has been used to verify and validate adaptation poke configurations. For instance, there is a conflicting issue between the pokes `add_assess_formative` and `add_assess_summative`, because they partly include the same question items. In addition, `add_assess_summative` is used to evaluate concepts which are not evaluated by `add_assess_formative`. Such conflicts could not be solved in analysis time, since the concrete components (i.e. question items) to be eventually integrated into the CDP were not known until design. All these activities are not automatic, but supervised by the instructional engineer. According to the feedback provided by the IMS LD ontology-based validator, the instructional engineer can iteratively refine the adaptation pokes and eventually decide which ones are valid and therefore incorporated to the CDP (Zarraonandia et al., 2007).

3.4 Learning implementation phase

The result of the design phase was the set of required adaptation components (i.e. units of learning, learning activities, learning objects and learning services) considered to be helpful in acquiring the desired competences. Should these exemplars exist in the repository, the transition from design to implementation would be straightforward. For instance, to implement navigation in the CDP design model when the `selfDirected` attribute is selected under `navigationService`, an IMS Simple Sequencing navigational service can be provided (Icodeon, 2007). Similarly, to implement the tutoring facility required by the `tutor`-based learner support feature, an IMS-LD-based monitor service *chunk* (Hernández-Leo et al., 2006b) can be used. Furthermore, such learning components and services can be easily replaced by others that provide the same function, which turns out as the task of the implementation phase.

However, sometimes there can be no suitable learning design components for the desired configuration of features, so new learning artefacts should have to be generated. Then, a configuration of LD *templates* or *patterns* must be provided to adapt the UoL (Hernández-Leo et al., 2006a). Such templates are abstract LD exemplars that should be completed thereafter, during the implementation phase. The goal is to obtain ready-to-run units of learning that build up the eventual CDP. In our example we will omit this step and suppose that suitable LD components were found after the design phase.

The components that eventually form the CDP adaptation are used to augment the repository of available learning components. These can be annotated with design rationales, mainly inferred from the learning analysis and design phases.

4 Conclusions and future work

Instructional Engineering methods are commonly used for design-time conception and adaptation of instructional material, such as units of learning, learning designs and competence development programmes. A generative, model-driven IE method is described in this paper and used to adapt a unit of learning to transform it into a CDPs. The generative framework consists of a feature model, a set of learning components and a meta-model-based transformational approach to eventually generate the learning artefact. Feature models are used to abstractly specify the desired adaptation, as well as to control allowed configurations of learning components. Since generative methods aim at modelling system families instead of individual systems, our method can be applied to adapt a given member of a learning system family, such as a CDP.

The generative IE method is based on analysing the CDP from different *levels of abstraction*, namely learning analysis, design, implementation and evaluation. For each phase, different models (i.e. the CDP feature model, the IMS-LD model, and the adaptation model) are used to represent the CDP with different abstraction levels. The distinction between learning *analysis* and *design* as separate phases of the method takes a step ahead in considering the possible dependencies between the CDP *concerns*. In software engineering methods, the resolution of design conflicts should be carried out as early as possible in the method. This is the aim of the feature modelling carried out in the learning analysis phase.

The generative framework defined in this paper has driven the transformations needed to map the abstract features into adaptations to the runnable UoL. As a first step to face up the learning *context* issues, configurations of feature model and IMS-LD ontology have been respectively used to determine what combinations of adaptations could be applied during analysis and design phases for a specific learning context. Nevertheless, the current selection of adaptations is far from automatic. The effort of automating the method outweighs the benefits if the CDP is not intended to be adapted for reuse in forthcoming learning settings, or if the learning process span is reduced. But if we often have the opportunity to reuse adapted versions of a CDP, or if a life-long learning process is needed, then the effort of automation is worth it. Nevertheless, it is not realistic to consider automation of the development process as the main purpose of the method. The ultimate goal is achieving the highest level of automation that is possible, whereas the generative approach acknowledges the possibility of different levels of automation. In spite of this, if we had considered the generative method as a process in which the computer automatically adapts a learning programme, and whose result must have a pedagogical sense, it becomes clear that the computer should have access to information about the design rationales used in adapting the CDP. These rationales are traced by the method as "after what goal and feature configuration did the designer choose this adaptation poke?" and are explicitly represented along with the outcome of the process.

Obtaining CDPs that are actually reusable for a number of learning contexts requires taking a step further and organizing adaptations as a system of patterns, whose application can be automated. This organization would help to select the adequate learning design pattern(s) and successfully manage their combination for a specific learning context, which is the first focus for our future work. A second point of further work is about the evaluation of adaptations. On the one hand, feature models enable experts or automated systems to analyse and find out in design time which component fails for a particularly desired CDP, or even measure which design decisions are more valuable for your learning assets. On the other hand, the adaptation model can be used for the run-time evaluation of the introduced adaptations. Although such capabilities have not been tested in the described case study, further work is required to feed the IE process with empiric evidences for the need of contextual adaptations to real learning situations.

Acknowledgements: This work is funded by the MODUWEB project (TIN2006-09768) from the Spanish Ministry of Science and Technology.

References

Amorim, M.A., Lama, M., Sanchez, E., Riera, A. and Vila, A.X. (2006). A Learning Design Ontology based on the IMS Specification. *Journal of Educational Technology & Society* 9(1), pp. 38-57

Brusilovsky, P. (2003) Developing adaptive educational hypermedia systems: From design models to authoring tools, In *Authoring Tools for Advanced Technology Learning Environment*, Dordrecht: Kluwer Academic Publishers, pp. 377-409.

Czarnecki, K., Antkiewicz, M. (2005). Mapping Features to Models: A Template Approach Based on Superimposed Variants, Proc. GPCE, pp. 422-437.

Czarnecki K. and Kim, C.H.P. (2005). Cardinality-based feature modeling and constraints: a progress report. *International Workshop on Software Factories*, 2005, San Diego, California.

Czarnecki, K., Kim, C.H.P. and Kalleberg, K.T. (2006). Feature models are views on onthologies, *Proceedings of 10th SPLC*, 2006, Baltimore, USA

Demirors, O., Demirörs, E., Tarhan, A. and Yildiz, A. (2000). Tailoring ISO/IEC 12207 for instructional software development, *Proceedings of the 26th Euromicro Conference*, 2000, pp. 2300-2308.

Derntl M. and Botturi, L (2006). Essential use cases for pedagogical patterns, *Computer Science Education*, 16(2), pp. 137-156.

Díez, D., Fernández, C. and Dodero J.M. (2006). Towards an effective instructional engineering analysis method, *Proceedings of the First European Conference on Technology Enhanced Learning*, 2006, Crete, Greece, LNCS 4227

Dodero, J.M., Sánchez-Alonso, S. and Frosch-Wilke, D. (2007). Generative Instructional Engineering of Competence Development Programmes, *Journal of Universal Computer Science* (to be published)

Fey, D., Fajta, R. and Boros, A. (2002): Feature Modeling: A Meta-Model to Enhance Usability and Usefulness, *Proceedings of the 2nd International Conference on Software Product Lines*, August, 2002, San Diego, CA, pp. 198-227.

Greenfield, J. and Short, K. (2004): *Software Factories. Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley Publishing

Herder E., Koesling A., Olmedilla D, Hummel, H. Schoonenboom, J., Moghnieh, A. and Vervenne, L. (2006) European lifelong competence development: requirements and technologies for its realisation, *Proceedings of the Workshop on Learning Networks for Lifelong Competence Development*, March, 2006, Sofia, Bulgaria

Hernández-Leo, D., Villasclaras-Fernández, E.D., Asensio-Pérez, J.I., Dimitriadis, Y., Jorrín-Abellán, I.M., Ruiz-Requies, I. and Rubia-Avi, B. (2006a). COLLAGE: A collaborative Learning Design editor based on patterns, *Educational Technology and Society*, 9(1), pp. 58-71.

Hernández-Leo, D., Harrer, A., Dodero, J.M., Asensio-Pérez, J.I. and Burgos, D. (2006b): Creating by Reusing Learning Design Solutions, *Proceedings of the 8th International Symposium on Computers in Education*, 2006, León, Spain, pp. 417-424.

Icodeon (2007). Icodeon Sequencing Engine. Web Services Pack, version 1.0.0, Tech. Report, Icodeon Ltd., Retrieved Feb 2, 2007 from http://www.icodeon.com/packs.do

Jovanović, J., Gasevic, D., Knight, C., Richards, G. (2006), Learning Object Context for Adaptive Learning Design, *Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Dublin, Ireland, 2006, pp. 288-292.

Koper, E.J.R. (2003). Combining re-usable learning resources and services to pedagogical purposeful units of learning. In A. Littlejohn (Ed.), *Reusing Online Resources: A Sustainable Approach to eLearning*, London: Kogan Page, pp. 46-59.

Leidig, T. (2001). L3-towards an open learning environment, *Journal on Educational Resources in Computing*, 1(1), article no. 7.

McAndrew, P., Goodyear, P. and Dalziel, J. (2006). Patterns, designs and activities: unifying descriptions of learning structures, *International Journal of Learning Technology*, 2(2-3), pp. 216-242.

Mizoguchi, R. and Bourdeau, J. (2000). Using Ontological Engineering to Overcome Common AI-ED Problems, *International Journal of Artificial Intelligence in Education*, 11(2), pp. 107-121.

Murray, T. (1996). Special Purpose Ontologies and the Representation of Pedagogical Knowledge, *Proceedings of the 1996 International Conference on Learning Sciences*, 1996, Evanston, Illinois, pp. 235-242.

Niles, I. and Pease, A. (2001). Towards a standard upper ontology, *Proceedings of the international conference on Formal Ontology in Information Systems*, 2001, Ogunquit, Maine, pp. 2-9.

Pankratius, V. and Vossen, G. (2005). Reengineering of educational material: A systematic approach, *International Journal of Knowledge and Learning*, 1(3), pp. 229-248.

Paquette, G., Aubin C. and Crevier F. (1999). MISA, A knowledge-based method for the engineering of learning systems, *Journal of Courseware Engineering*, 2, Fall, pp. 63-78.

Paquette, G. (2004). *Instructional Engineering in Networked Environments*. Pfeiffer-Wiley.

Polsani, P.R. (2003). Use and abuse of reusable learning objects, *Journal of Digital Information*, 3(4).

Robson, R., Collier, G. and Muramatsu, B. (2004). Reusability Guidelines for Collections. ACM/IEEE Joint Conference on Digital Libraries Workshop, Tucson, AZ.

Sloep, P. Hummel, H. and Manderveld, J. (2005). Basic design procedures for e-learning courses, in Koper, E.J.R. & Tattersall, C. (Eds.) *Learning Design: A Handbook on Modelling and Delivering Networked Education and Training*, Berlin-Heidelberg: Springer Verlag, pp. 139-160.

Tennyson, R.D. and Barro, A.E. (1995). *Automating Instructional Design: Computer-based Development and Delivery Tools.* Springer

Uden, L. (2003). An engineering approach for online learning, *Journal of Distance Engineering Education*, 1(1), pp. 63.

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

Zarraonandia, T., Dodero, J.M., Fernandez C. and Aedo I. (2007). Iterative Design of Learning Processes, in B. Fernández-Manjón and J.M. Sánchez-Pérez (Eds.) *E-Learning: from Theory to Practice*, Springer, 2007.

Footnotes

[1] Traditionally, Spanish Universities' curricula only provided one final evaluation for each 4-month course. Nowadays, this is being changed, due to the European policies for higher education, to a continuous assessment scheme that eventually guides to achieving Bologna's declaration objectives. This change is being implemented gradually, first on newcomers' courses.