Journal of Universal Computer Science, vol. 13, no. 9 (2007), 1213-1233
submitted: 15/8/06, accepted: 1/3/07, appeared: 28/9/07 © J.UCS

Generative Instructional Engineering of Competence Development Programmes

Juan Manuel Dodero
(Universidad Carlos III de Madrid, Spain
dodero@inf.uc3m.es)

Salvador Sánchez-Alonso
(Information Engineering Research Unit, University of Alcala, Madrid, Spain
salvador.sanchez@uah.es)

Dirk Frosch-Wilke
(Faculty of Business Management, University of Applied Sciences, Kiel, Germany
dirk.frosch-wilke@fh-kiel.de)

Abstract: Competence development programmes are collections of units of learning and learning activities used to increase the overall effective performance of a learner within a certain task. The definition of a competence development programme is fairly complex and subject to variability, depending on the available learning units and components. Some instructional engineering approaches have been successfully used to create courseware by the combination of existing learning resources within a systematic and iterative method. In this work, a generative, model-driven engineering approach is used to create and adapt competence development programmes from families of available learning components, such as units of learning, learning designs, and learning services. The process begins from the statement of the learning goals as feature models, and carries out a number of transformations from the analysis model down to learning designs and implementation components. However, shared definitions for competence-related terms and computational semantics are essential in this effort. In this paper, ontologies are proposed as a means to that end. In particular, the transformations between models are defined with the help of a general competence ontology.

Keywords: Competence Development Programmes, Learning Design, Instructional Engineering
Categories: D.2.2, K.3.0, I.2.4

1 Introduction

The success of companies and other organizations is often linked to the positive contribution of human resources. However, the same human factor can also be the cause of the company's failure. In most contexts, examining the competences of the workforce (and using the results to decide how to properly allocate human resources to the "right places and tasks") is essential to the achievement of both individuals and organizations. This measure is in fact vital for the organization reaching long-term success.

The term competence is interpreted in several ways in the literature. One of the most complete definitions of the term defines competence as "a specific, identifiable,

definable, and measurable knowledge, skill, ability and/or other deployment-related characteristic (e.g. attitude, behaviour, physical ability) which a human resource may possess and which is necessary for, or material to, the performance of an activity within a specific business context" [Allen 2006]. However, registration and development of competences and abilities for employees and functions is a complex task. Recent research in the context of industrial processes reinforces the assumption that the assessment of human performance demands new methods for representing individual competences. In this regard, [Zülch and Becker 2006] state that:

"It is [...] necessary to examine in detail competences acquired of the people as well as the various requirements for necessary competences. On this subject however there appears to be a lack of methods with which individual competences during planning and decision-making can be taken into account. A fixed terminology and suitable concepts for their representation in analysis and planning processes will be necessary. However, in these fields certain bases are still missing."

This affirmation, which could be easily translated to other fields, is the major motivation for this paper. Using it as a point of departure, we will later examine computer-aided methods that can be of help in the first stages of assignment planning, as well as the formalisms necessary to establish a shared terminology on competences. The use of ontologies as a way of stating shared, formal definitions of competence-related concepts will provide the foundations upon which these methods are based.

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

Even though 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 part of the so-called Instructional Engineering discipline [Paquette 2004], aimed at increasing the degree of automation of the instructional design process. This paper describes a generative instructional engineering method used to create and adapt CDPs, which considers the methodological complexity of the process along the engineering lifecycle.

The rest of this paper is structured as follows: in [Section 2] a definition of competence is provided and related to current standards. [Section 3] explains the required formalisms and ontologies used to represent competences and to enable the definition of CDPs. In [Section 4], an instructional engineering approach to create CDPs is described and put in the framework of current instructional engineering practices, followed by conclusions and future work in [Section 5].

2 Competence: Definition and Standards

At present, several different definitions of the concept of "competence" coexist. Although most agree on a few core characteristics, it is interesting to provide a brief discussion about some of the most closely related to this work.

The notion of competence is often considered as a "placeholder" for knowledge, skill, abilities, and "other characteristics" [Sicilia 2005]. However, this view can be judged as an excessive oversimplification of the many facets of the use of the term [Hoffmann 1999]. In a general sense, a competence can be defined as "an underlying characteristic that leads to successful performance, which may include knowledge and skills as well as bodies of knowledge and levels of motivation" [Rothwell 2006]. Another broad definition is that included in the IMS-RDCEO Best Practices and Implementation Guide [Cooper and Ostyn 2002c]: "All classes of things that someone, or potentially something, can be competent in". Some other authors believe that competences encompass more than just knowledge and skill, as they "focus on what is unique about individuals doing the work rather than what people must know or do to perform the work alone" [Rothwell 2006].

In order to support and use effectively the link between competence and CDPs, there is need to provide reusable definitions of competences, among different systems [CEN 2005]. In the rest of this section, the most prominent approaches to competence standardization are studied, which are aiming to provide a solution to this problem. It should be remarked that, as it has been stated earlier, most agree on the core characteristics of competences, even though all include their own definitions and consequently refer to the term competence from their own perspective.

2.1 IMS-RDCEO

The IMS consortium (http://www.imsglobal.org) provides a specification for competences called "Reusable Definition of Competence or Educational Objective (RDCEO)". IMS-RDCEO defines an information model for describing, referencing, and exchanging definitions of competences, primarily in the context of online and distributed learning. This specification aims to provide the means for formally representing the key characteristics of a competence, independently from its use in a particular context.

IMS RDCEO competence data may include a definition of the competence, evidences of the competence, information about its context and the scale (i.e. proficiency on a predetermined scale). It defines a set of elements of information, in five different categories, that can be used to define a competence: identifier (subdivided into catalog and entry), title, description, definition (subdivided into model source and statement) and metadata (subdivided into RDCEO schema, RDCEO schema version and additional metadata).

IMS-RDCEO is expected to promote common understanding of competencies that can be used in competence development (learning and career development), or in specifying learning pre-requisites and learning outcomes. The complete specification consists of three documents:

- IMS-RDCEO Information Model [Cooper and Ostyn 2002a], that includes the complete description of the main elements of the specification: semantics, structure, data types, value spaces, multiplicity, and obligation.
- IMS-RDCEO XML Binding [Cooper and Ostyn 2002b], just one example of the bindings that might use the information model.
- IMS-RDCEO Best Practices and Implementation Guide [Cooper and Ostyn 2002c], a set of rules about the application of both the Information Model and the XML Binding, as well as examples.

However, although IMS-RDCEO is a widespread description model for competences there still exist some open issues in its description capabilities as described by [Karampiperis *et al.* 2006].

2.2 HR-XML

The HR-XML (http://www.hr-xml.org/) is an independent, non-profit consortium, whose main aim is to enforce e-commerce and inter-company exchange of human resources data within a variety of business contexts. Starting as an initiative in Northern America, today also European and Asian chapters promote the distribution of the proposals in other regions and countries. The main effort supported by this consortium is the development of standardized XML vocabularies for human resources, as well as standards for staffing and recruiting, compensation and benefits, and training and work force management. Major companies such as Addeco, Cisco Systems, PeopleSoft GmbH, IBM, Microsoft, and many others are currently members of the HR-XML Consortium.

Up to the present, the HR-XML Consortium has produced a library of more than 100 interdependent XML schemas that define the data elements for particular HR transactions, as well as options and constraints governing the use of those elements. One of the schemas provided by the HR-XML Consortium is the denominated *Competences Recommendation*. This set of recommendations about competences allows "the capture of information about evidence used to substantiate a competence and ratings and weights that can be used to rank, compare, and otherwise evaluate of the sufficiency or desirability of a competence" [Allen 2006].

The competences schema is particularly relevant to processes involving the rating, measuring, comparing, or matching an asserted competence (for example, a skill claimed in a resume) against one that is demanded (for example, a skill required in a job description). This fact, added to the fact that this schema is intended as a module that can be incorporated within broader process-specific schemas, facilitates its use outside the HR domain as a general-purpose competence schema, and consequently its integration in diverse frameworks. The only requirement for those frameworks is, of course, the use of some kind of competence management.

[Fig. 1], taken from [Allen 2006], depicts the components of a competence after what is stated in the HR-XML recommendation. For each competence, a number of elements of information are defined, but also the structure and information of the competence evidences and weights, among other information.

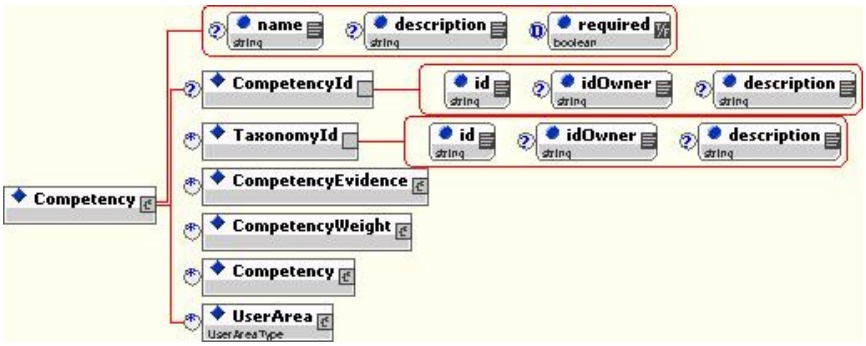

Figure 1: Components of a HR-XML competence

The definition of a competence, according to this schema, is shown in the following example, taken from [Allen 2006]:

```
<Competency name="Oral Comprehension"
description="The ability to listen to and understand information and
ideas presented through spoken words and sentences">

<CompetencyId id="1.A.1.a.1"/>

<TaxonomyId id="O*NET" idOwner="National O*Net Consortium"
description="Occupational Information Network"/>

<CompetencyWeight type="x:Importance">
  <NumericValue maxValue="100" minValue="1">65</NumericValue>
</CompetencyWeight>

<CompetencyWeight type="x:Level">
  <NumericValue maxValue="100" minValue="1">57</NumericValue>
</CompetencyWeight>
</Competency>
```

An interesting feature is that HR-XML can also be used as a wrapper of an RDCEO record by making use of a Uniform Resource Name (URN).

3 The Need for Formalisms to Represent Competence-Related Concepts

We have already mentioned the need of a fixed terminology of competence-related concepts, as expressed by [Zülch and Becker 2006], for their representation in analysis and planning processes. Previous section detailed current standardization efforts to attain this goal. However, current standards for the definition, sharing and exchange of competences insist in the construction of models of competences mostly intended for human interpretation, lacking semantic interoperability.

In fact, this important shortcoming of current standards for the description of competences, i.e. the lack of formalization, hampers the automated management of

competence descriptions and makes difficult the implementation of complex features. Specifications dealing with competences, such as HR-XML Competences and RCDEO, while useful for data interchange, do not provide the required computational semantics because the information they contain is, in its current form, intended for human consumption only. Present practices result in data lacking machine-understandable characteristics, which makes their use in Semantic Web environments difficult if not impossible.

The solution to this problem consists in formalizing concepts in ontology languages such as OWL [Dean and Schreiber 2004] or WSML [WSMO 2005], thus bestowing upon competence management the benefits of the Semantic Web vision [Berners-Lee *et al.* 2001]. An obvious advantage is the potential to integrate competence management into broader frameworks like the Knowledge Management Lifecycle of the KMCI model [McElroy 2003], as sketched by [Sánchez-Alonso and Frosch-Wilke 2007]. Besides addressing the need for a shared set of competence-related terms and definitions, a formal representation of concepts in an ontology provides machine-understandable data, and facilitates the automated management of the information in competence records and descriptions.

3.1 Using Ontologies to Formally Represent Competence-Related Terms: the Case of the LUISA Project

The LUISA Project[1] addresses the development of a reference semantic architecture for the major challenges in the search, interchange and delivery of learning materials in a service-oriented context. The implementation of the architecture is put into practice in two real-world case studies (one in the Academic e-Learning and other in the Industrial e-Training), converting existing learning resources to the ontological representation. This serves both as a proof of concept for the architecture, and as a way to analyze empirical usage findings in real-world contexts. LUISA, which stands for *Learning Content Management System Using Innovative Semantic Web Services Architecture*, manages the use of learning materials (in the form of learning objects) helped by standard IEEE LOM-compliant [IEEE 2002] metadata descriptions. However, learning object metadata are often XML centric and have weak or no support for semantic web expressions. This was the main motivation for engineering an ontology of competence-related concepts as part of the project. The terms and relationships of this ontology, called Generic Competence Ontology (GCO), were inspired by the directives and guidelines provided by the standardization efforts described in [Section 2] but also by previous work by Sicilia [Sicilia 2005].

In the context of LUISA, the two case studies mentioned are based on the concept of competence, even though from different standpoints. Yet, providing a general purpose schema for competences has been approached to increase the re-usability and flexibility of the resulting technologies. The core competence ontology finally engineered [see Fig. 2] is aimed at being applied to automated human resource management scenarios in general and to learning scenarios in particular.

[1] http://luisa.atosorigin.es/index.htm

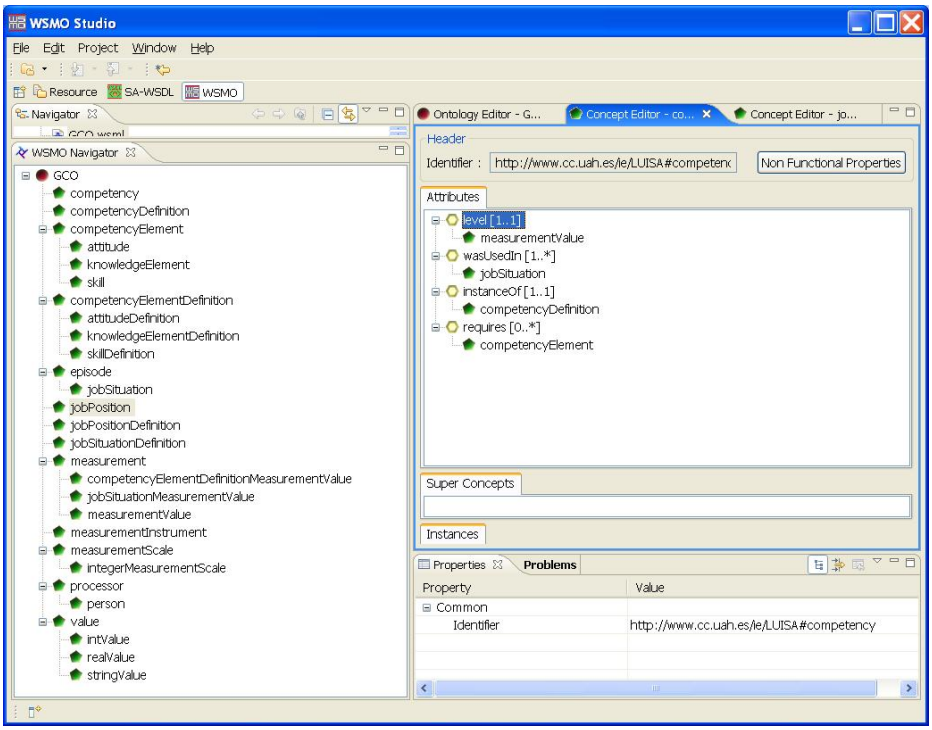

Figure 2: Editing the LUISA GCO in the WSMO Studio[2] ontology editor

As contemplated in LUISA, the notion of competence is linked to the concept of human performance, which according to the model of Rummel [Rothwell and Kazanas 1992] encompasses several elements:

1. The work situation, as the origin of the requirement for action that puts the competence into play.
2. The individual's required attributes (knowledge, skills, and attitudes) to act in a given work situation.
3. The response, which is the action itself.
4. The consequences or outcomes (in fact the results of the action), which determine if the standard performance has been met.

Finally, individuals usually receive feedback depending on the success or failure of their actions.

[2] http://www.wsmostudio.org/

3.2 The LUISA Generic Competence Ontology (GCO) in Detail

The LUISA GCO separates actual competences, associated to particular individuals, from the definition of competences as stereotypes. The `competency` concept represents a discrete competence of an individual, which is generically represented as a `processor` to provide room for software systems also capable of exhibiting competences. In GCO, the competences are a characteristic of a `processor`, so that the `isAbleToPerform` relationship between instances of the terms `competency` and `processor` can be understood as a composition. The `level` attribute in competences is used to denote that some kind of `measurementScale` is required for competences. In WSML, the ontology language used to engineer the LUISA GCO, this information is represented as follows:

```
concept competency
     level impliesType  (1 1) measurementValue
     wasUsedIn impliesType  (1 *) jobSituation
     instanceOf impliesType  (1 1) competencyDefinition
     requires impliesType competencyElement

concept processor
     isAbleToPerform impliesType  (1 *) competency

concept person subConceptOf processor
     holds impliesType  (1 *) jobPosition
```

Several elements influence competences, such as knowledge elements, skills, and attitudes. A `knowledgeElement` is defined as "what is conveyed by usable representations" [Holsapple and Joshi 2004], referring to some discrete mental structures that can be represented in information artefacts like books and web pages. A `skill` is considered "an ability that has been acquired by training", whereas an `attitude` is defined as "a complex mental state involving beliefs and feelings and values and dispositions to act in certain ways". These three concepts have been modelled in GCO as subclasses of `competencyElement`:

```
concept competencyElement
     instanceOf impliesType  (1 *) competencyElementDefinition
     level impliesType  (1 1) measurementValue

concept knowledgeElement subConceptOf competencyElement
     instanceOf impliesType knowledgeElementDefinition

concept attitude subConceptOf competencyElement
     instanceOf impliesType skillDefinition

concept skill subConceptOf competencyElement
     instanceOf impliesType skillDefinition
```

These definitions allow for a clear separation about three types of traits that represent different aspects of competence. For example, a person may have the *knowledge* about the internal components of a certain machine, since she has studied several diagrams about it. This is different from having the *skill* of using it efficiently. In fact, the knowledge about the internals of the device may not be necessary for its

proper usage, and on the contrary, knowing the internals does not guarantee that this person is able to use the machine efficiently. In addition, *attitudes* represent elements not necessarily connected to specific knowledge or skills. For example, having good negotiation skills does not always entail that an employee would have the attitude to reach a consensus in meetings. From an ontological perspective, attitudes are mostly domain independent, while knowledge items and skills are not. Examples are "service orientation" or "attentive to details" attitudes that are equally applicable to employees, irrespective of the industry. Some skills are also of a generic nature, like "persuasion" or "negotiation" but many others refer to concrete elements or artefacts that are specific of the industry. Typical examples are "Java programming skill", "Linux administration", "Repairing Ford engines", and the like.

Competences are put into play in concrete `jobSituations`, which can be considered as a kind of `episode` in the life of the organization that occurs at a specific moment in time. The `requires` attribute of the concept `jobSituation` models the requisites (in the form of necessary competences) to hold a `jobPosition`, as for instance, a skill required in a job description. [Fig. 3] shows the full set of attributes of the concept `jobSituation`.

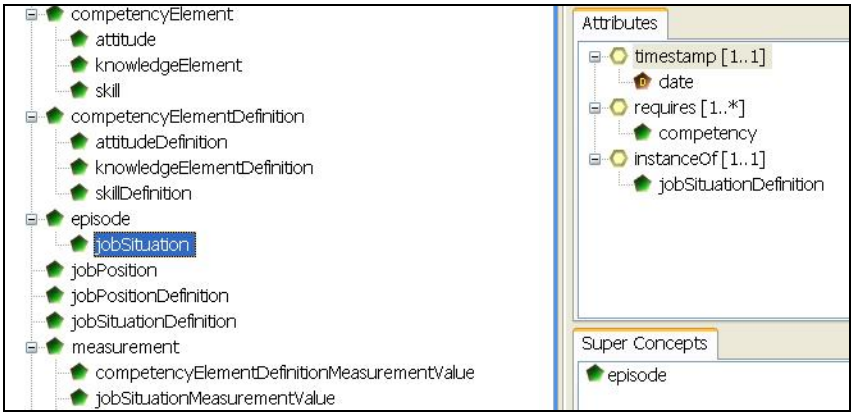

Figure 3: Attributes of the concept `jobSituation`.

Competences and `jobSituations` are connected to their respective "definition" elements. These definitions are used to represent stereotypical competences and job contexts, so that they can be used to describe, for example, job position characterizations in human resource selection processes, or even as a way to state the needs of a project. To describe work situations in terms of required competences, each `jobSituationDefinition` requires a number of competences as defined in `competencyDefinitions`. In WSML:

```
concept jobSituation subConceptOf episode
     timestamp impliesType  (1 1) _date
     requires inverseOf(wasUsedIn) impliesType  (1 *) competency
     instanceOf impliesType  (1 1) jobSituationDefinition
```

```
concept jobPosition
     instanceOf impliesType  (1 1) jobPositionDefinition

concept jobSituationDefinition
     requires impliesType  (1 *) competencyDefinition
```

Competence specifications are implicitly related by the relationships among competence components. For example, if a competence `c1` is considered to require some knowledge `k1`, then the competence implicitly requires the knowledge of any `k1` pre-requisite knowledge. This is represented through the `prerequisite` relationship which provides support for modelling knowledge trees:

```
concept knowledgeElementDefinition
   subConceptOf competencyElementDefinition
   prerrequisite impliesType knowledgeElementDefinition

concept skillDefinition subConceptOf competencyElementDefinition
     prerrequisite impliesType knowledgeElementDefinition
```

Measurement scales for competences can also be of a diverse nature. In the LUISA GCO, a `measurement` is connected to competences as an elaboration of the `level` attribute of the `competency` concept. Measurements are always related to a given `measurementScale`, and usually some `measurementInstruments` associated to such scales are available (e.g. questionnaires or interviews). Any `jobPosition` is described in terms of competence definitions by specifying a given `measurementLevel`, connected to the scale in which the level is expressed. Several types of scales and their associated measurements can be defined, each scale compulsorily providing some definitions that act as constraints on the description of the measurements. Specific scales can be defined as an instance of the term `integerMeasurementScale`:

```
concept measurement
     scaleUsed impliesType  (1 *) measurementScale
     currentValue impliesType  (1 1) value

concept measurementScale
     instrumentUsed impliesType measurementInstrument

concept measurementInstrument

concept integerMeasurementScale subConceptOf measurementScale
     zeroLevel impliesType  (1 1) _integer
     topLevel impliesType  (1 1) _integer
```

Finally, it is worth noting how the GCO ontology deals with flexibility in competence specifications. This is in fact approached in two ways. First, a competence definition is made up of several `competencyElements`, which are specialized into the three subclasses mentioned earlier: `skill`, `attitude` and `knowledgeElement`. And secondly, the schema allows for incomplete definitions of competences. Remarkably, a `competence` is entirely defined only if this is explicitly indicated by using a Boolean attribute. On the contrary, a competence can

be partially defined either if it is defined as a primitive competence (i.e. its elements are not defined) or if the described components do not define the competence completely.

4 Instructional Engineering of Competence Development Programmes

Instructional Engineering is defined as a discipline supporting 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 IE discipline is rooted in [Tennyson and Barro 1995], who described earlier attempts to automate instructional design. More recently, [Sloep *et al.* 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 strongly influenced by software engineering practices. CEM [Uden 2003] divides the courseware engineering life cycle in 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). MISA [Paquette *et al.* 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). Another attempt to automate instructional design is the ISDP [Demirörs *et al.* 2000], which 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 different levels of abstraction, a widely used technique to manage complexity. They also define a number of models to depict different aspects of the learning system. Nevertheless, the way in which dependencies between orthogonal views (or models) of the system are handled is rather limited. Aspect-oriented techniques, oriented to manage cross-cutting concerns in software engineering, have been purposed to engineer [Ateyeh and Lockemann 2006] and re-engineer [Pankratius and Vossen 2005] courseware and educational material. Since the separation of concerns is a relevant principle to modelling the multiple perspectives of a CDP —which are far more complex than simple courseware—, we adopted it in our systematic IE method. In what follows, we will first explain where the methodological complexity of designing a CDP lies, to later describe our IE method.

4.1 Methodological Issues

During the creation of a CDP, the instructional engineers can provide the input to the IE process with different *abstraction levels*, targeted on different instructional *contexts*, and affected by different *concerns*. On one hand, the input can be specified with a certain level of abstraction. 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 "difficult and long assessment." Furthermore, she can model the questions in the quiz according to their difficulty level, or let a computer-assisted quiz service composition tool do it.

On another hand, a CDP is normally useful only for a specific instructional context. Instructional designers should not try to design universally valid programmes. A major difficulty to achieve reusability is that learning objects [Polsani 2003] or learning designs [Koper 2003] are aimed to function in different instructional contexts. For instance, the design of a course on Descriptive Statistics should not be the same for Electronic Engineering students than for Social Science students. Moreover, once the course is designed, should it also work for Computer Science studies? These context-related issues have to be modelled as well.

Finally, the learning goals of a CDP are not usually unique. Instead, they can be decomposed in a number of goals (i.e. extensively), which in addition can be attained with a certain accuracy level (i.e. intensively). Both the extension and the intensity of the learning goals are contemplated as the *concern* of creating the CDP. For instance, during a project-oriented course on software engineering, the concern can be related to learning the planning and estimation processes, or it can be focused only on managing the development method and tools. Yet, both goals can be intended, but the required levels of fulfilment can be different for each different goal. The alternative goals can be depicted as a selectable hierarchy of learning objectives, which can yield quite different programmes.

These issues are handled by regular IE methods by defining a number of design models, which are used to manage the diversity of design models and abstraction levels. To solve the context issue, *learning design patterns* have provided parameterized templates that can be adapted to the instructional requirements or goals [McAndrew *et al.* 2006]. To model and structure such requirements, *goal-oriented* approaches have been defined for pattern maintenance and application [Derntl and Botturi 2006]. Nonetheless, dealing with a set of design models entails further questions on how and when merging them to yield the final CDP. In the following, we introduce a generative IE method, in which the analysis phase is directed by the instructional goals to determine the desired features of the CDP. This method uses patterns to map the desired features onto a concrete learning design. It is based upon defining and using meta-models to drive the transformations needed to generate an utilizable set of learning components in the implementation phase.

4.2 Generative Domain Model

As first step of common generative modelling practices, a *generative domain model* must be defined to serve as the framework of the method. 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. competence-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, cross-cutting aspects of design. The generative IE process encourages the efficient use of system models in instructional design by engineering *families* of learning components (i.e. members) that eventually may 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: a means of specifying family members, the implementation components from which each member can be assembled, and the configuration knowledge used to generate a finished member from a number of member specifications.

The specification of instructional requirements that starts up the IE process operates at a higher level of abstraction, and can be done according to different domain models. Since such models are not completely unrelated from each other, model mappings and transformations needed to assemble learning components are not straightforward. In our method, that issue is undertaken by metadata and ontology-based transformations, which are actually semantic meta-models of the configuration knowledge. Implementation components that build up a CDP are constituted by available units of learning, learning designs, learning activities, learning objects and learning services [Koper 2003]. For simplicity, we will refer to these as *learning components* or simply *components*. According to our approach, these components must be appropriately annotated to enable model-based transformations. This is done by using the ontological representation of competences described in [Section 3].

Finally, a higher-level specification of family members must be defined. These are provided by *feature models*, structured collections of eligible features that include formalized definitions of features and composition rules (i.e. rules that determine which combinations of features are valid, as well as rules that indicate whether selecting certain features require the presence of other features or not). Feature models allow building service-level agreements on a CDP, so that instructional designers (or even learners) can select the instructional goals and the required level of fulfilment of such goals in a concrete learning artefact. These decisions will be traceable from the initial learning objectives to the final assessments.

4.3 Instructional Engineering Method

Having all these rationales in mind, the generative IE method can be summarized in the following iterative phases, which form the so-called engineering workflow:

- The *learning analysis* phase consists in selecting a consistent set of features (or deriving it from the learning goals definition), ensuring the application of restrictions on feature combinations. Then an analysis model is produced. This phase makes use of ontologies such as the one discussed in [Section 3]
- During the *design* phase, the delivered analysis model is transformed into a number of abstract learning designs that guarantee the service-level agreement for the CDP derived from the analysis phase. This is done by annotating learning designs and activities with elements derived from the competence ontology. Eventually, a design model is generated.

- In the *implementation* phase, those concrete implementation components that better suit to the abstract learning design model are found. If such implementation components are not available, they should be built to cover the demand.
- The *evaluation* phase closes the iteration loop, compiles the design rationales occurred during the previous phases, and annotates the implementation components for further improvements in the method.

In the following, the instructional engineering method is exemplified and applied to designing a concrete CDP intended to attain a given university position for a learner (e.g. becoming an assistant student). For the sake of illustrating how the CDP is built, a full iteration of the method is followed along its main phases.

Since the instructional engineering process is goal-driven, a number of potential goals must be provided at the beginning. For simplicity, we considered only three goals of a student during her university career, which can be expressed in WSML as follows:

```
goal _"http://example.org/myUniversity/enrolAsNewcomer"
   nonFunctionalProperties
      dc#title hasValue "Goal of enroling myUniversity as a newcomer"
      dc#type hasValue _"http://www.wsmo.org/TR/d2/v1.2/#goals"
   endNonFunctionalProperties
goal _"http://example.org/myUniversity/engage"
   nonFunctionalProperties
      dc#title hasValue "Goal of engaging myUniversity as a non first-
                         year student"
      dc#type hasValue _"http://www.wsmo.org/TR/d2/v1.2/#goals"
   endNonFunctionalProperties
goal _"http://example.org/MyUniversity/becomeAssistantStudent"
   nonFunctionalProperties
      dc#title hasValue "Goal of becoming an assistant student in
                         myUniversity teaching activities"
      dc#type hasValue _"http://www.wsmo.org/TR/d2/v1.2/#goals"
   endNonFunctionalProperties
```

The selection of one goal triggers the analysis phase, which uses a *feature-oriented domain analysis* approach [Kang *et al.* 1990]. [Fig. 4] depicts the feature model used for the specification of CDP family members. The 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]—. Such features are mapped to abstract learning design components in the forthcoming design phase.

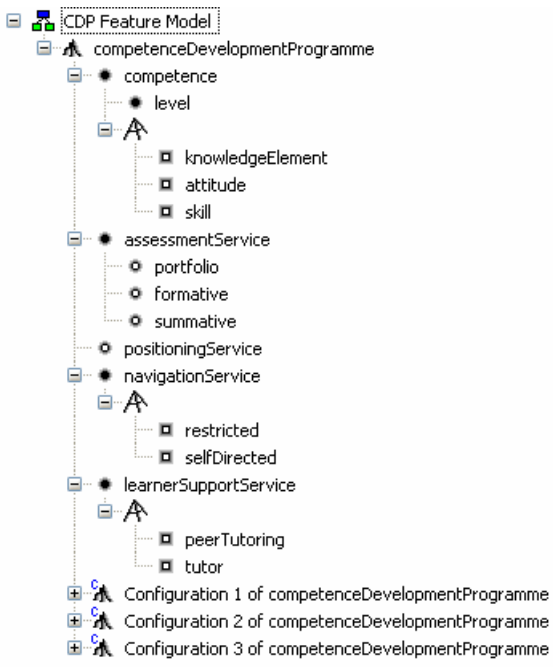

Figure 4: A feature model for the specification of CDP family members

Goals are translated into feature configurations, which are group of features commonly selected altogether. The `becomeAssistantStudent` goal is mapped to the feature configuration number 3 [see Fig. 5], which consists of the following elements: a given competence of type `skill`; no positioning service required; navigation service allowed with any option; and learner support service based on peer tutoring. 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.

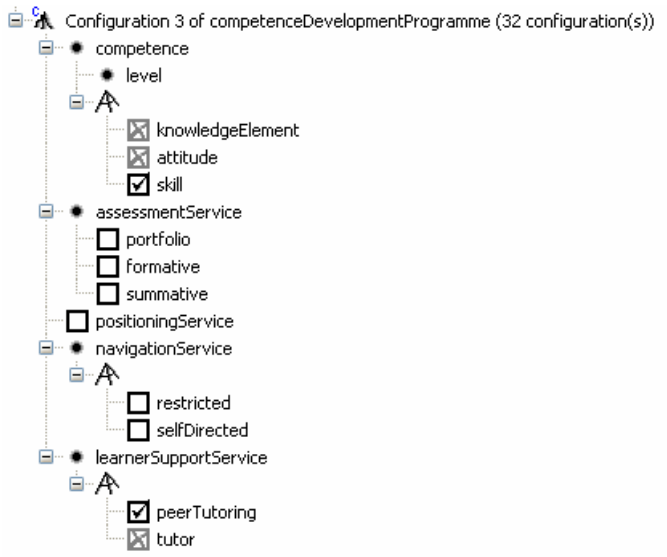

Figure 5: A configuration of CDP to achieve the goal `becomeAssistantStudent`

The design phase involves the selection of appropriate components to map these abstract services from a repository of learning components. Since *reference attributes* are out of the scope of feature modelling [Czarnecki *et al.* 2006], a more expressive way to model this mapping is needed. This is carried out with the help of the ontology below, based on the General Competence Ontology (GCO) presented in [Section 3].

```
namespace { _"http://myuniversity.edu/myUniversityCompetences#",
 gco _http://www.cc.uah.es/ie/LUISA# }

ontology _"http://myuniversity.edu/myUniversityCompetences#"

concept profession
     professionTitle impliesType _string

concept engineeringStudent subConceptOf gco#person
     hasName impliesType _string
     hasID impliesType _integer
     hasActualSkills impliesType gco#skill
     hasActualAP impliesType scaledAP
     haActualCompetences impliesType myUniversityCompetence
     hasPosition impliesType position

concept abilityForEngineeringProfession
     aPCode impliesType _string

concept myUniversityCompetenceDefinition subConceptOf
gco#competenceDefinition
     compCode impliesType _string
     isUsedInAbilityForP impliesType abilityForEngineeringProfession
```

```
concept scaledAP
     instanceOf impliesType abilityForEngineeringProfession

concept proficencyLevel subConceptOf gco#measurement

concept myUniversityCompetence
     instanceOf impliesType myUniversityCompetenceDefinition
     compCode impliesType _string

concept studentPosition subConceptOf gco#jobPosition
     requiresReqAP impliesType scaledAP
     requiresCom impliesType myUniversityCompentency
     positionID impliesType _integer
     positionDesignation impliesType profession

concept assistantStudent
     instanceOf impliesType studentPosition
     posCode impliesType _string
```

In the current example, the feature model outcoming from the analysis phase provides the input to the ontology-based inference engine, which determines a set of design components that help a regular `engineeringStudent` to reach an `assistantStudent` position with the desired set of `competences`.

However, there can be multiple design pathways to achieve the same competence. The feature-oriented analysis CDP yields the number and range of possible feature configurations, which is helpful to refining the feature model, narrowing the range of design possibilities, and eventually converging to the next step of the method.

During the analysis and design phases, if the selected goals and features lead to conflicting or non-compatible configurations the ontology would provide the appropriate information to correct such cases. For instance, in the case of an already `assistantStudent` which selects the goal `enrolAsNewcomer` in the same `myUniversity`. Nevertheless, conflicting features or design decisions can occur several times, which may cause to have to go forth and back between analysis and design steps, either in the same or in successive iterations of the engineering workflow.

The design outcome is a set of required implementation components assumed to be helpful in acquiring the desired competences: the knowledge in the ontology guarantees that the chosen components are the best among the available. Sometimes, however, there can be no suitable learning components for the desired configuration of features, so new learning artefacts will have to be generated from scratch. In that case, a configuration of abstract learning *exemplars* is provided by the method, consisting of learning design *templates* that must be *completed* (according to the terminology of [Hernández-Leo *et al.* 2006]) during the implementation phase. The objective is to obtain ready-to-run units of learning that will constitute the CDP. In the example, IMS LD *patterns* [Hernández-Leo *et al.* 2005, McAndrew *et al.* 2006] can be generated and completed thereafter.

The result of the design phase is the set of required implementation 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. Similarly, to implement the peer tutoring facility required by the `peerTutoring` attribute selection, either a readily available one-on-one communication tool or a web-based forum can be used. Although the selection of one tool over others becomes irrelevant with respect to the feature model considered in this example, the choice should take into account other analysis features or design constraints that might be introduced in further iterations of the method (e.g. include some usability features)

The components eventually forming the CDP are used to augment the repository of available learning artefacts. These are annotated with design rationales, mainly inferred from the learning analysis and design phases. The competence ontology of our example can be used to annotate the outcome CDP; to trace back, for instance, that the peer-to-peer tutoring support component present in the CDP was selected in order to facilitate the goal of `becomeAssistantStudent`. This might be helpful in future instructional decisions. However, the evaluation should be enriched with results taken from *a posteriori* analytic surveys delivered to the team of instructional engineers, or even compiling opinions from the learners.

5 Conclusions and Future Work

A generative instructional engineering method has been described to develop and adapt CDPs, which consists of a feature model, a set of learning components and an ontology-based transformational approach to eventually generate the CDP. Feature-oriented analysis is used to specify, in an abstract manner, the desired features of the programme and then launch the instructional engineering workflow. The engineering lifecycle iterates through the main phases of analysis, design, implementation and evaluation, where a number of transformations between models are carried out. This is driven by an ontology of competences, namely the LUISA General Competence Ontology (GCO). As a result, the generative instructional engineering method aims at modelling learning system families instead of modelling individual learning systems.

Feature models also enable experts or automated systems to analyse and find out which learning component fails for a particularly desired CDP attainment, or even to measure which past design decisions were more valuable for given learning assets. To enable these possibilities, further work should focus on incorporating design rationales in the evaluation phase, as well as carrying out prototype-testing in real learning design situations. The compilation of design rationales opens the way to feeding back the engineering process, which in turn gives a chance to re-design instructional practices.

Acknowledgements

This work is partly funded by the 6th EU framework IST project LUISA (FP6-027149); and by the MODUWEB project (TIN2006-09768) from the Spanish Ministry of Science and Technology.

References

[Allen 2007] Allen, C. (Ed.): "HR-XML Competences (Measurable Characteristics)" Retrieved January 1, 2007, from: http://ns.hr-xml.org/2_4/HR-XML-2_4/CPO/Competences.html

[Ateyeh and Lockemann 2006] Ateyeh, K., Lockemann, P. C.: "Reuse- and Aspect-Oriented Courseware Development", Educational Technology and Society, 9, 4, 1996, 95-113.

[Berners-Lee 2001] Berners-Lee, T., Hendler, J., Lassila, O.: "The Semantic Web" Scientific American, 284, 5, 2001, 34-43.

[Brusilovsky 2003] Brusilovsky, P.: "Developing adaptive educational hypermedia systems: From design models to authoring tools", Authoring Tools for Advanced Technology Learning Environment, Kluwer Academic Publishers, Dordrecht, 2003, 377-409.

[CEN 2005] European Committee for Standardization: "A European Model for Learner Competencies", CWA 15455, Retrieved January 8, 2007 from: http://www.ni.din.de/sixcms/media.php/1377/CWA15455.pdf?backend_call=true

[Cooper and Ostyn 2002a] Cooper, A., Ostyn, C. (Eds.): "IMS Reusable Definition of Competency or Educational Objective - Information Model, Version 1.0, Final Specification", Retrieved January 1, 2006, from:
http://www.imsglobal.org/competencies/rdceov1p0/imsrdceo_infov1p0.html

[Cooper and Ostyn 2002b] Cooper, A., Ostyn, C. (Eds.): "IMS Reusable Definition of Competency or Educational Objective - XML Binding, Version 1.0, Final Specification", Retrieved January 1, 2006, from:
http://www.imsglobal.org/competencies/rdceov1p0/imsrdceo_bindv1p0.html

[Cooper and Ostyn 2002c] Cooper, A., Ostyn, C. (Eds.): "IMS Reusable Definition of Competency or Educational Objective - Best Practice and Implementation Guide, Version 1.0, Final Specification", Retrieved January 1, 2006, from:
http://www.imsglobal.org/competencies/rdceov1p0/imsrdceo_bestv1p0.html

[Czarnecki and Antkiewicz 2005] Czarnecki, K., Antkiewicz, M.: "Mapping Features to Models: A Template Approach Based on Superimposed Variants", Proc. GPCE, 2005, 422-437.

[Czarnecki et al. 2006] Czarnecki, K., Kim, C. H. P., Kalleberg, K. T.: "Feature models are views on ontologies", Proc. 10th SPLC, Baltimore, USA, 2006, 21-24.

[Dean and Schreiber 2004] Dean, M., Schreiber, G. (Eds.): "OWL Web Ontology Language Reference", W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-ref/

[Demirörs et al. 2000] Demirörs, O., Demirörs, E., Tarhan, A., Yildiz, A.: "Tailoring ISO/IEC 12207 for instructional software development", Proc. 26th Euromicro Conference, 2000, 2300-2308.

[Derntl and Botturi 2006] Derntl, M., Botturi, L.: "Essential use cases for pedagogical patterns", Computer Science Education, 16, 2, 2006, 137-156.

[Herder et al. 2006] Herder, E., Koesling, A., Olmedilla, D., Hummel, H., Schoonenboom, J., Moghnieh, A., Vervenne, L.: "European lifelong competence development: requirements and technologies for its realisation", Proc. Workshop on Learning Networks for Lifelong Competence Development, Sofia, Bulgaria, 2006, 88-92.

[Hernández-Leo et al. 2005] Hernández-Leo, D., Villasclaras-Fernández, E. D., Asensio-Pérez, J. I., Dimitriadis, Y., Jorrín-Abellán, I. M., Ruiz-Requies, I., Rubia-Avi, B.:

"COLLAGE: A collaborative Learning Design editor based on patterns", Educational Technology and Society, 9, 1, 2005, 58-71.

[Hernández-Leo et al. 2006] Hernández-Leo, D., Harrer, A., Dodero, J. M., Asensio-Pérez, J. I., Burgos, D.: "Creating by Reusing Learning Design Solutions", Proc. 8th SIIE, León, Spain, 2006.

[Hoffmann 1999] Hoffmann, T.: "The meanings of competency", Journal of European Industrial Training, 23, 6, 1999, 275-286.

[Holsapple and Joshi 2004] Holsapple, C. W., Joshi, K. D.: "A formal knowledge management ontology: Conduct, activities, resources, and influences", Journal of the American Society for Information Science and Technology, 55, 7, 2004, 593-612.

[IEEE 2002] IEEE: "Draft standard for Learning Object Metadata", IEEE 1484.12.1-2002, 2002, available at: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf

[Kang et al. 1990] Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S.: "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Report CMU/SET-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1990.

[Karampiperis et al. 2006] Karampiperis, P., Sampson, D.G., Fytros, D.: "Lifelong Competence Development: Towards a Common Metadata Model for Competencies Description - The Case Study of Europass Language Passport", Proc. 6th ICALT, 2006, 677-681.

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

[McAndrew et al. 2006] McAndrew, P., Goodyear, P., Dalziel, J.: "Patterns, designs and activities: unifying descriptions of learning structures", International Journal of Learning Technology, 2, 2-3, 2006, 216-242.

[McElroy 2003] McElroy, M. W.: "The new knowledge management: Complexity, learning, and sustainable innovation", KMCI Press, Butterworth-Heinemann, Boston, MA, 2003.

[Pankratius and Vossen 2005] Pankratius, V., Vossen, G.: "Reengineering of educational material: A systematic approach", International Journal of Knowledge and Learning, 1, 3, 2005, 229-248.

[Paquette et al. 1999] Paquette, G., Aubin, C., Crevier, F.: "MISA, A knowledge-based method for the engineering of learning systems", Journal of Courseware Engineering, 2, fall, 1999, 63-78.

[Paquette 2004] Paquette, G.: "Instructional Engineering in Networked Environments", Pfeiffer-Wiley, 2004.

[Polsani 2003] Polsani, P. R.: "Use and abuse of reusable learning objects", Journal of Digital Information, 3, 4, 2003.

[Rothwell 2006] Rothwell W. J.: "A Report on Workplace Learner Competencies", Retrieved January 1, 2006, from: http://www.ilpi.wayne.edu/files/roth_present.pdf

[Rothwell and Kazanas 1992] Rothwell, W. J., Kazanas, H.: "Mastering the instructional design process", San Francisco, CA: Jossey-Bass, 1992.

[Sánchez-Alonso and Frosch-Wilke 2007] Sánchez-Alonso, S., Frosch-Wilke, D.: "An ontological representation of competences as codified knowledge", M. A. Sicilia (Ed.),

Competences in Organizational E-Learning: Concepts and Tools, Hershey, PA: Information Science Publishing, 2007 (in press).

[Sicilia 2005] Sicilia, M. A.: "Ontology-based competency management: Infrastructures for the knowledge-intensive learning organization", M. D. Lytras and A. Naeve (Eds.), Intelligent learning infrastructures in knowledge intensive organizations: A semantic Web perspective, Idea Group, Hershey, PA, 2005, 302-324.

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

[Tennyson and Barro 1995] Tennyson, R.D., Barro, A.E.: "Automating Instructional Design: Computer-based Development and Delivery Tools", Springer, 1995.

[Uden 2003] Uden, L.: "An engineering approach for online learning", Journal of Distance Engineering Education, 1, 1, 63-64.

[WSMO 2005] WSMO: "The Web Service Modeling Language WSML, WSML Final Draft 5", Report D16.1v0.21, October 2005. http://www.wsmo.org/TR/d16/d16.1/v0.21/20051005/

[Zülch and Becker 2006] Zülch, G., Becker, M.: "Computer-supported competence management: Evolution of industrial processes as life cycles of organizations", Computers in Industry, 2006, doi:10.1016/j.compind.2006.09.012 (to be published)