International Journal of Engineering Education Vol. 32, No. 2(B), pp. 956–968, 2016
Printed in Great Britain
0949-149X/91 $3.00+0.00
© 2016 TEMPUS Publications.

A Software Engineering Framework to Assist Instructors in Eliciting Course Requirements*

AZEDDINE CHIKH and JAWAD BERRI
Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia.
E-mail: {az_chikh, jberri}@ksu.edu.sa

This paper proposes a software engineering framework that aims to assist instructors in expressing their courses' requirements. The framework allows an instructor to elicit his know-what and know-how knowledge in teaching a course through a set of learning activities that are the building blocks of e-courses. The instructor's expertise is elicited through a graphical user interface that provides the necessary tools for producing a course description which is then converted automatically into a course specification allowing software engineers and programmers to implement it as an e-course. We anticipate that this framework would be very helpful for instructors to express their course requirements in a systematic and convenient way. In this paper we present the framework and show by means of a case study related to a systems analysis and design course, how this framework is used to produce the e-course specifications from the instructor requirements.

Keywords: software systems requirements engineering; education engineering; elicitation; specification; course requirements; unit of learning; activity; metadata

1. Introduction

The availability of Web 2.0 technologies promoting interaction and collaboration has enabled new forms of services to satisfy user needs in terms of information, entertainment and social networking [1]. The learning field has benefited a great deal from these technologies enabling the creation of e-courses in all fields. The digital learning scene engendered by Web 2.0 technologies is becoming collaborative, participatory, and non-linear. In this context Web 2.0 Learning is emerging as a new paradigm of modern education [2]. The widespread use of learning management systems and the prevailing of computer devices such as mobile smart phones and tablet computers along with the availability of wireless networks in urban spaces have urged instructors to transit from the traditional face to face learning towards more interactive forms of e-learning [3]. Lots of options exist nowadays to move to e-learning, the goal is to motivate and interest learners through effective participation in class [4]. However, it is not always easy for instructors to embark in this transition and not all of them are able to make the suitable choice with regard to the technology that best fits the context of their courses. Although lots of efforts have been deployed in developing environments and standards to ease e-learning adoption, there are still barriers that prevent instructors from developing systematically e-courses to support their teaching. A major obstacle is that these technologies require an understanding of the functionalities that are offered and a certain level of computer literacy. While most of efforts in education engineering have been focusing on using technical tools to support instructors in teaching and to facilitate the learning process for learners, very little attention has been dedicated to support instructors in the development of new courses through an engineering methodology that frees the instructor from the complexity of the technology and lets him focus on the pedagogical side. The development of e-courses remains a real burden for instructors as it requires the creation of the learning material and it necessitates from instructors to have software engineering skills to be able to accomplish the different phases of e-courses' development.

Education engineering can help a great deal in defining the different actors' roles in e-learning. Teachers are the appropriate actors to provide the know-what and the know-how which represents the expertise related to teaching a subject. The know-what is the necessary knowledge related to a domain and the know-how is about the way this knowledge is dispensed to learners. Teachers should be solicited to express their expertise in a seamless manner using a well-established methodology to transfer their expertise into a e-course. On the other side, software engineers who master the technologies and who are aware about the technical complexity of existing tools should be requested to codify the expertise of teachers into e-courses. The development of e-courses needs to be considered as an engineering process with resources, actors, and a final product to be developed through a succession of steps that is likely to be similar to the software development life cycle [5]. Through this vision instructors are system users who express their needs and software engi-

* Accepted 1 September 2015.

neers are the developers of the e-course. Software requirements engineering (SRE) is a discipline that can help a great deal with the analysis phase in the development life cycle of e-courses. SRE is the process of identifying the users' needs and documenting these in a form that can be evaluated, communicated and subsequently implemented [6]. This process includes three major phases: elicitation, specification, and validation of software requirements.

This paper proposes an activity based framework which aims to elicit requirements in the field of education engineering that is meant to be used by instructors to help them express their needs in the development of e-courses. This framework is based on the concept of Learning Activity which is the building block of instruction. Then the requirements are translated into specifications which are used by software developers to implement e-courses. We anticipate that this framework would be very helpful for instructors to express their course requirements in a systematic and convenient way and will allow them to focus on the pedagogical side of course development. We show in a case study related to a course on Systems Analysis and Design, how this framework is applied.

2. Background

2.1 Learning design

Instructional Design is the systematic development of instructional specifications using learning and instructional theory to ensure the quality of instruction [7]. It is the entire process of analysis of learning needs and goals and the development of a system to meet those needs. It includes development of instructional materials and activities; and tryout and evaluation of all instruction and learner activities.

The IMS Learning Design (IMS-LD) [8] specification intends to represent the learning design (LD) of units of learning (UoL) in a semantic, formal and machine interpretable way. A UoL can be any instructional or learning event of any granularity, e.g. a course, a workshop, a lesson or an informal learning event. A LD is defined as the description of the learning process that takes place in the UoL. The key principle in learning design is that it represents the learning activities and the support activities that are achieved by different actors (learners, instructors, tutors) in the context of a UoL. These activities can refer to different learning objects (LO) that are used in activities, and it can refer to services that are used to collaborate and to communicate in the learning process. The IMS-LD specification is developed to meet some specific requirements:

1. *Completeness*: The specification must be able to fully describe the learning process in a UoL, including references to the digital and non-digital LOs and services needed during the process. This includes: (a) Integration of the activities of both learners and staff members. (b) Integration of resources (LOs and communication/collaboration services) used during learning. (c) Support for both single and multiple user models of learning. (d) Support for mixed mode (blended learning) as well as pure online learning.
2. *Pedagogical expressiveness*: The specification must be able to express the pedagogical meaning and functionality of the different data elements within the context of an LD. While it must be sufficiently flexible to describe LDs based on all kinds of pedagogies, it must avoid biasing designs towards any specific pedagogical approach.
3. *Personalization*: The specification must be able to describe personalization aspects within an LD, so that the content and activities within a unit of learning can be adapted based on the preferences, portfolio, pre-knowledge, educational needs and situational circumstances of users. In addition, it must allow the designer, when desired, to pass the control over the adaptation process to the learner, a staff member and/or the computer.
4. *Compatibility:* The specification must enable LDs to use and effectively integrate other available standards and specifications where possible, such as the IMS (imsglobal.org) and IEEE LTSC (ltsc.ieee.org) specifications. The IMS Learning Design specification consists of several components. First of all it consists of a conceptual model (an ontology) for the description of teaching-learning processes which is expressed as an UML model. In essence the model says that learners perform a set of learning activities using LOs and services (activity environment) in order to attain some explicit or implicit learning objectives. As a result of the activities, the learners produce outcomes (e.g. reports, forum/wiki contributions, etc.) that subsequently can be used by others in their learning or support activities (e.g. an instructor can provide feedback to a report written by a learner). Instructors, other staff members or peers can perform support activities to help learners when needed. The design can be static or adaptive, taken into account the existing competencies, needs and circumstances of the persons involved.

2.2 Activity based learning

Activity based learning is a paradigm of learning that focuses on the activity as the granular unit of analysis. It is a milieu that nourishes peer to peer learner interaction, collaboration and social communication [9]. A learning activity stimulates cognitive and social processes by placing learners into the context of the real world allowing them to communicate and search interdisciplinary domains [10]. This paradigm fits very well with e-learning which is inherently modular and interactive. Modularity of e-learning systems is expressed through units of learning that are independent and autonomous allowing them to be exchanged and reused. Interactivity is materialized by all the technology enhanced learning environments that promote communication and data exchange and provides hypermedia resources for navigation [11]. Reusing learning activities is an important feature of activity based learning systems to exchange units of learning seamlessly. Successful implementation of this feature faces two challenges namely: the activity's granularity and the technology used to represent activities. The granularity is related to the activity content; it is the level of finesse at which the activity is described. Granularity needs to be characterized uniformly through a unified model to allow activity exchange between learning courses. For instance, an activity can be coarse such as "Prepare a presentation about the greenhouse effect" or it can be fine focusing on a very simple task such as "Add slide numbers to the presentation". On the technical side, reusing learning activities across different learning systems and platforms presupposes that activities are represented using common standards to allow seamless search, exchange and integration. This is possible if these environments are equipped with: (i) the structural components required for the support of learning activities, (ii) common ways for representing learning scenarios and (iii) the required educational systems' services [12].

An interesting implementation of the concept of activity in e-learning is the system LAMS (Learning Activity Management System) [13]. LAMS is a tool for designing, managing and delivering online collaborative learning activities. It is designed as a visual authoring environment for creating sequences of learning activities. LAMS is very interesting as it is based on activities which are the basic learning chunks for the development of learning courses. As an implementation of the learning activity concept, it provides a complete environment including various tools and technical facilities to be used as a standalone system or to be combined with other existing learning management systems [14]. However, LAMS has a couple of limitations which are: (i) the system is oriented towards the design and management of learning sequences based on activities. In other words, LAMS user (author or instructor) is drove to start with the design of learning activities and sequence of activities presupposing that he has done the basic and essential homework that is to analyze the requirements of the course in terms of know-how where knowledge and concepts are expressed, organized and formalized; (ii) LAMS user is bound to express the user's know-how using the taxonomy of activities available in the system through the "Activity Toolkit". Although the taxonomy is diversified including a significant set of activities, these cannot encompass all types of learning activities that a user can think of; (iii) Sequencing of activities is basic as it does not include the richness of relationship semantics such as prerequisite and composition. Moreover, LAMS is not an open system for coding or extending existing features.

3. Related work

Requirements engineering has been used in e-learning to solve strategic issues. Indeed the incorporation of e-learning into the curricula of traditional higher education institutions involves many complex issues such as strategic management decisions, strategic information technology plans, change management to enhance the stakeholders' willingness to participate, and course development [15]. The problem is how to successfully identify the key processes for the adoption of e-learning within some organization? The solution to this issue could be brought by business consultants where the most efforts would be spent on requirements elicitation. Authors in [16] describe a methodology for requirements elicitation of traditional higher educational environments in order to gain a comprehensive understanding of its key processes. The authors claim that such an understanding might be helpful in the strategic planning and e-learning implementation that is both successful in achieving learning outcomes, and are also completely integrated into all key processes with positive reception from stakeholders. The methodology uses the basic underlying theoretical principles of Software Requirements (RE). The methodology was designed to accelerate the RE process and accordingly reduce the initial development cost.

Scenarios have been advocated as a mean of improving RE yet few methods or tools exist to support scenario-based RE. Authors in [17] reported a method and software assistant tool for scenario-based RE that integrates with use case approaches into object-oriented development. The method and operation of the tool are illustrated with a financial system case study. Scenarios are used to represent paths of possible behavior through a use case, and these are investigated to elaborate requirements. Scenario paths are automatically generated from use cases. Scenarios are validated by rule-based frames which detect problematic event patterns. Authors in [18] proposed a supporting collaborative requirements elicitation framework using Focus Group Discussion Technique. A prototype of Focus Group Discussion for requirements elicitation tool, FGD-RElicit assist all stakeholders (the course developer and the school instructors) during the requirements elicitation process.

Scenarios are used in [19] directly in the design where the authors show how the Learning Design (LD) Editor needs to perform two sets of functions: allowing to create pedagogic scenarios and defining the flow of activity along with the various branching conditions. These are used either as a single design or as a template. There is a quite distinct requirement that calls for an LD Editor to be able to populate a design with specific resources and services. In the former case there is a role of an "expert" who defines a pedagogic scenario, while in the latter case it is often the instructor who writes the scenario with what is needed for a particular session. As suggested in LD [20], modeling a learning design is a three-stage process: informal modeling in nature language, semi-formal modeling in UML activity diagrams, and formal modeling in XML. Modeling a learning design starts with elicitation. The goal of elicitation is to acquire all information needed to describe the desired learning design. Such process information involves the objectives and context of the learning design, the learning content and facilities used in the learning process, the principal entities such as roles and activities, and any relationships among them in terms of workflow. It is expected to describe behavior features of the process (e.g., under which conditions an activity can start or complete), if necessary and possible. In LD, it is suggested to describe process information in a structured manner.

4. Unit of learning development

In our approach of education engineering we propose a global method for developing a unit of learning UoL (which can be a course, a module, a lesson, etc.). This method is based on the Model Driven Architecture (MDA) [21] which provides an approach that uses models to represent UoL development from requirements to business modeling to technology implementations. MDA is a software engineering framework that promotes visualizing, storing, and exchanging software designs and models which can then be used for the production of documentation, acquisition specifications, system specifications, technology artifacts and executable systems. There, they can be accessed repeatedly and automatically transformed by tools into schemas, code skeletons, test harnesses, integration code, and deployment scripts for various platforms [22]. The development of a UoL includes three main phases: UoL Analysis, UoL Design, and UoL Implementation.

1. *UoL Analysis*: this phase starts with the elicitation step where the instructor provides a metadata based description of his UoL. This description will be later translated into a UoL specification, which needs to be validated.
2. *UoL Design*: this phase aims to design a UoL. It aims to create the IMS Content Package: a formal model represented in LD (Learning Design) has to be specified in the form of XML. The creation of a UoL involves the creation of a learning design and also the bundling of all its associated resources, either as files contained in the unit or as web references, including assessments, learning materials and learning service configuration information. As a result, a packaging mechanism is used to pack the learning design and its associated files into a single container the IMS Content Package.
3. *UoL Implementation*: the IMS Content Package will serve as the main input for this phase. This requires a learning management system or some dedicated software environment that is able to parse the IMS Content Package.

5. Analysis of a unit of learning

The MDA approach decouples the requirements analysis phase, which goal is to express the requirements of UoLs, from the design and implementation phases. This approach prevents the instructor from being biased by any design method or tool and avoids turning the requirement analysis phase into a pure implementation and coding task. Also the proliferation of learning management systems and the popularity and availability of Web 2.0 technologies in e-learning deviates sometimes instructors from their core pedagogical function that is to facilitate, prompt and guide the learning process towards expressing their needs in terms of what the technologies offer and what are its limitations. This work focuses on the requirement analysis phase as it facilitates the expression of the instructor's expertise in an intuitive way through activities description. Figure 1 illustrates the three steps of the requirements analysis life cycle for a given UoL.

5.1 Elicitation

In an activity-based learning where the activity is the

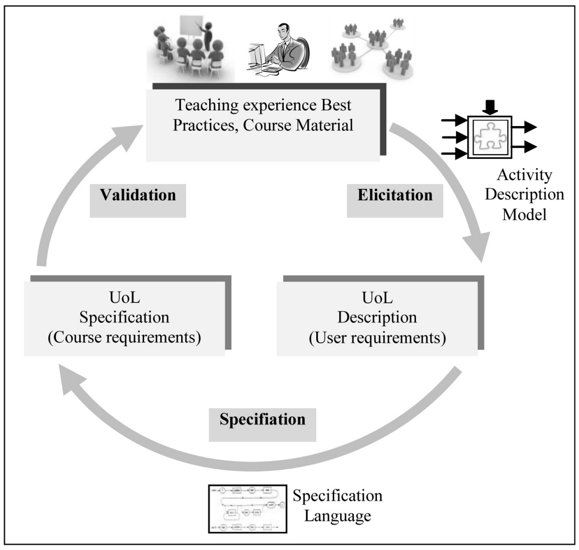
Teaching experience Best Practices, Course Material
Activity Description Model
Validation
Elicitation
UoL Specification (Course requirements)
UoL Description (User requirements)
Specifiation
Specification Language

Fig. 1. Requirements analysis of a UoL.

building block of instruction, elicitation consists in describing the activities included in the UoL and their sequencing. We define a learning scenario as a series of learning activities to be achieved by actors; using learning objects, tools and services. It reflects the instructor's pedagogy in delivering the UoL.

The elicitation step handles the question of how instructors' needs could be represented as a UoL description (user requirements). It relies mainly on the instructor's knowledge, teaching experience, best practices in one side and the UoL material (course description; resources/ references. . .) in another side. This step is meant to help instructors to write user requirements using the activity description model (ADM).

ADM is useful because (1) it forces instructors to think about how learners will learn while they are exposed to the UoL; (2) it also helps building a testing dataset at the end of the UoL implementation phase. The ADM, which is based on activities, allows to describe the UoL at several levels as shown in Figure 2:

1. *UoL scenario (level 0):* the instructor describes first the main activities of the UoL. Examples of main activities could be: doing a course project; summarizing a book chapter; simulating a process; comparing two design approaches.
2. *Activity structure (levels 1 to n):* is considered as a recursive breakdown of the UoL scenario. Indeed, each main activity from UoL scenario should be further detailed into an activity structure composed of sub-activities. Examples of sub-activities of the main activity "doing a course project" could be: analyzing the problem; designing the solution; and implementing the solution. Recursively, each sub-activity could in turn be broken down into smaller fine grain activities giving rise to a structure at a lower granularity level.

Each learning activity in the ADM (Fig. 3) is identified with a unique identifier (ID) and is expressed in a narrative way. For each activity the instructor specifies a set of metadata: (i) Instructional metadata such as the activity type; the learning situation; the prerequisite knowledge; the objectives and outcomes; (ii) Quality metadata related to quality such as usability; reliability; maintainability, and performance; (iii) Components metadata identifying the required activity's components: learning objects; services/tools; and actors; and (iv) Relationship metadata defining the relationships between activities.

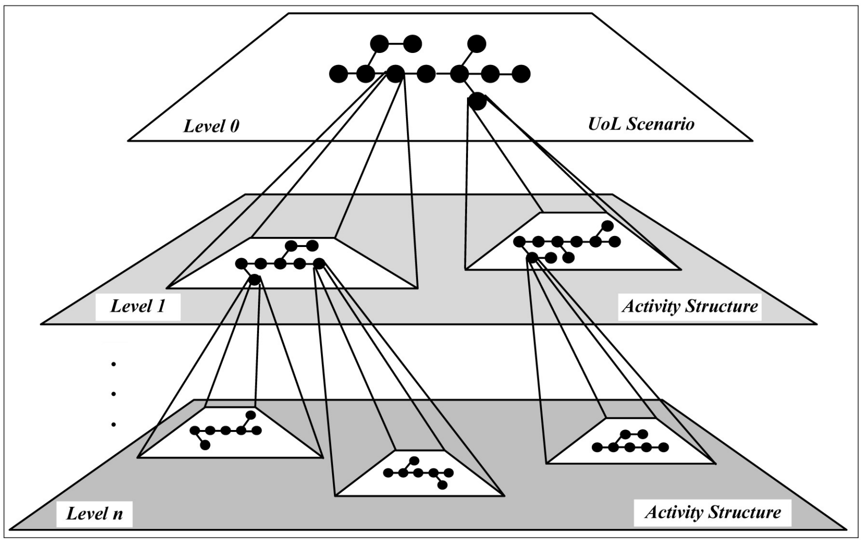

Fig. 2. Multi-level UoL description.

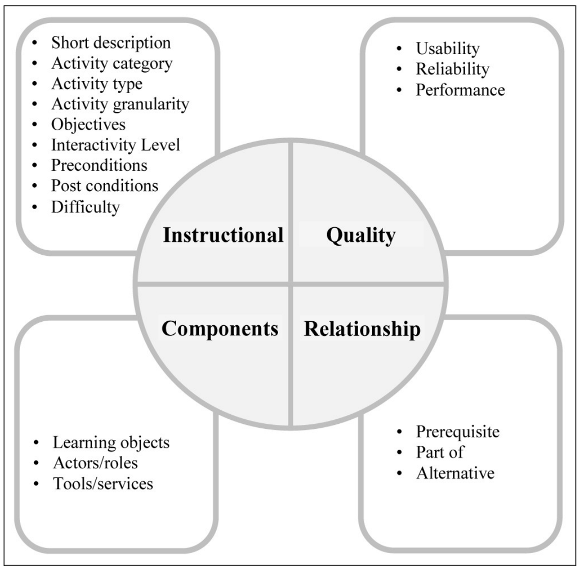

Fig. 3. The ADM structure

- *Instructional metadata*
  1. Short description.
  2. Activity category: individual, collaborative or cooperative.
  3. Activity type: learning activity or support activity.
  4. Activity granularity: main activity (0), atomic (level of activity structure).
  5. Objectives: Learning outcomes of the learning activity.
  6. Preconditions: what must be true before the learning activity begins.
  7. Post conditions: what must be true when the learning activity is completed?
  8. Interactivity Level.
  9. Difficulty.

These two last attributes are adopted from LOM metadata

- *Quality metadata*
  1. Usability;
  2. Reliability;
  3. Performance. . .
- *Components metadata*
  1. Description of learning objects used as resources in the learning activity.
  2. Description of actors/roles (learner, tutor. . .) involved in the learning activity.
  3. Description of tools/services used in the learning activity
- *Relationship metadata*

If a learning activity invokes or includes other activities then these will be linked to it by a set of possible relationships defined within the framework:

1. The present activity is *a prerequisite* for another one.
2. The present activity is *a part* of another one.
3. The present activity is *an alternative* for another one.

5.2 Specification

The UoL description (User requirements) represented in ADM will further be exploited to determine the UoL specification (Course requirements). The derivation of course requirements from user requirements is automatic. One advantage of this derivation is that it will contribute to better structuring course requirements. The UoL specification accommodates the required activities and their ordering as well as the potential constraints related to the involved actors, and the available resources and services. It includes functional and non-functional requirements:

1. *Functional course requirements:*
An example of functional course requirements for a Systems Analysis and Design course is given below:

FR1: Learners in this UoL should achieve the following learning activity: *Add the short description of the activity.*

2. *Non-functional course requirements:*
We look at each of the previous functional course requirements in turn, and consider whether there are any non-functional requirements.

They encompass the following categories:

- *Instructional requirements*:

An example of instructional requirements for a Systems Analysis and Design course is given below:

NFR1: Interactivity in FR1 shall be: *Add the interactivity level of the activity from ADM*

- *Quality requirements*: usability; reliability; performance. . .

Examples of Quality requirements for a Systems Analysis and Design course are given below:

(a) *Usability requirement:*

NFR2: Usability in FR1 shall be: *Add the usability of the activity from ADM*

(b) *Reliability requirement:*

NFR3: Reliability in FR1 shall be: *Add the reliability of the activity from ADM*

(c) *Performance requirement:*

NFR4: Performance in FR1 shall be: *Add the performance of the activity from ADM*

- *Components requirements:*

Examples of components requirements for a Systems Analysis and Design course are given below:

(a) *Resources requirements:* learning objects;

NFR5: The activity in FR1 shall make use of the following Learning objects: *Add the list of LOs from ADM*

(b) *Services requirements:* tools/devices and other services;

NFR6: The activity in FR1 shall make use of the following services: *Add the list of services from ADM*

(c) *Actors requirements:* learners; tutors, other staff. . .

NFR7: The activity in FR1 shall involve the following actors: *Add the list of actors from ADM*

- *Relationships requirements:*

An example of Relationships requirements for a Systems Analysis and Design course is given below:

NFR8: The activity in FR1 is a prerequisite to: *Add the list activities from ADM*

5.3 Validation

This step aims to ensure that the UoL specification obtained in step2 reflects well the real instructors' needs initially expressed as UoL description represented in ADM (step1). ADM is very efficient during this step to test and validate the user requirements.

6. Framework architecture

In order to assist the instructor in eliciting the course requirements, the proposed framework provides a graphical user interface which offers an ADM template for activity description, a set of sample activities and activity structures, as well as a set of UoL scenarios. Course requirements elicitation is a cognitive activity that consists in expressing the know-what and know-how related to a course. The know-what is the necessary knowledge about the course content including the concepts, the competencies and skills that need to be grasped by learners. The know-how is the way to dispense the knowledge to learners which includes the pedagogical methods and the technologies used. The framework has been designed so that to be used by any instructor and hence does not require a high level of IT literacy.

The framework's architecture presented in Figure 4 includes four modules: (1) UDE (UoL Description Editor); (2) UDB (UoL Description Browser); (3) UDM (UoL Description Manager); (4) USG (UoL Specification Generator).

6.1 UoL description editor

The UoL Description Editor (UDE) module, which is based on the ADM, aims to create and update user requirements, during the step 1 of the requirements analysis life cycle. It allows creating new activities and updating existing ones. Activities' related data is stored within a repository. Each activity of the UoL is obtained by instantiation of the concept of activity in ADM. The UDE module offers a convenient interactive working space for an efficient elicitation of user requirements. It allows instructors writing a short description of the activity, specifying its different activity components (resources, services and actors), describing its instructional characteristics as well as its quality features. Once activities are finalized the instructor

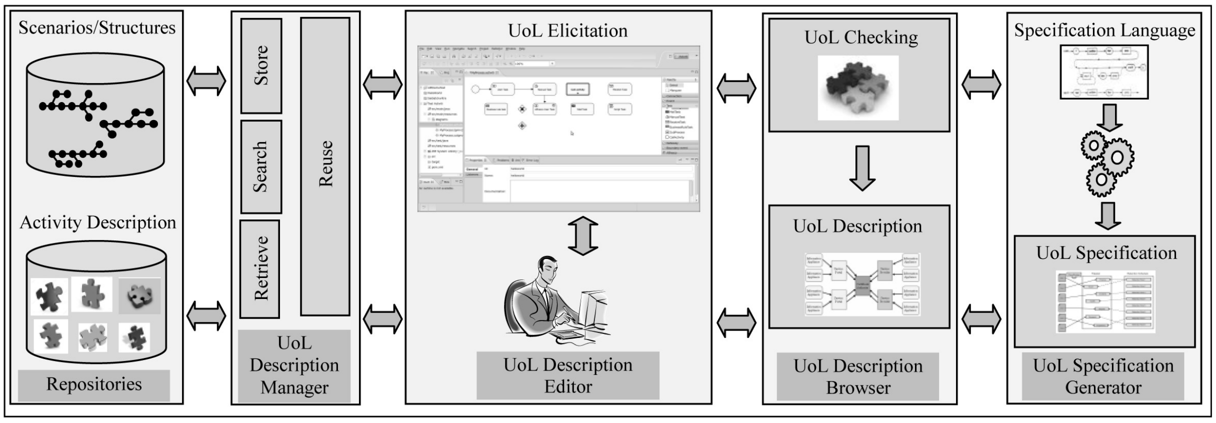

Fig. 4. The framework architecture.

can define links between activities that define the possible sequence between them.

Activities in the repository can be reused into other UoLs. In such case they are retrieved and probably updated to adapt to the new UoL. The resulting activities are stored in the repository as new instances. Reusing activity instances to develop a UoL is a bottom up approach where the elicitation starts by the activity instances that are then adjusted to fit with the UoL at hand. This approach is very convenient for the development of compatible UoLs, for example belonging to the same course, and may save significant time and efforts since it does not require to start over from scratch for every UoL. The second approach is to describe an activity by using the ADM requiring a new description from scratch. Both approaches might be used in the elicitation of a given course. Hence for developing the same UoL, the instructor can describe activities either from scratch using the ADM, or by reusing existing activities from the repository.

6.2 UoL description browser

The UoL Description Browser (UDB) module aims to create and update UoL scenarios or activity structures. This process follows a top-down approach. When the instructor has finished expressing the main activities of the UoL in the UDE, he starts building the UoL scenario (an interconnected set of the main activities). An activity structure (an interconnected set of sub-activities) could be built for each main activity of the previous level. The resulting sub-activities could be recursively broken down into others.

We leave some freedom to the learners by making the activity structure very flexible. Indeed we adopt the point of view of the authors in [23] who consider a learning activity as intrinsically evolutionary and unpredictable. The instructor can only build an a priori description of what he wants it to be; he must then have the means of modifying and adapting it dynamically, whereas it is being carried out. This supposes that he has at his disposal the means of observing the activity.

The UoL scenario or the activity structure integrates all the activities (sub-activities) defined in the UDE working space considering all the constraints defined in the ADM. The integration is done by the UDB which helps the instructor to create a new scenario or structure for a given UoL or activity. When building a UoL scenario or activity structure, the instructor makes use of activities or sub-activities using the drag and drop technique offered by the GUI. He can choose the suitable synchronization mode parallel or sequential. Whether in the creation of a UoL scenario or an activity structure, the UDB ensures that the activity ordering satisfies the constraints described by the relationship metadata category in ADM. Finally the UoL scenario and the activity structure can then be displayed to the user. They can be modified by the instructor in case he wishes to reorder the activities sequence or simply add or remove some activities. This cycle can be repeated as long as the user is not satisfied with the current UoL scenario or activity structure. When the user agrees he can choose to accept the UoL scenario or activity structure which is then saved in the repository. UoL scenarios and activity structures are the building blocks of a course and hence can be adapted and reused.

Reusing these blocks in new similar UoLs is very attractive and does not require lots of developments. New functions however need to be added to the graphical user interface to allow their interactive adaptation in the UDE working space.

The browsing function allows accessing the user requirements through dynamic hyperlinks. In fact the repository is dynamically converted into a hyper document.

6.3 UoL description manager

The UoL Description Manager (UDM) module handles the access to the existing user requirements. It includes an interactive Graphical User Interface (GUI) that facilitates retrieving, browsing, deleting, and disseminating user requirements.

User requirements, which are represented in ADM with a unique identifier, can be listed in the GUI and can be reordered according to a set of criteria extracted form ADM metadata. Hence they allow the instructor to browse them in a convenient way. The Search function allows the instructor to retrieve user requirements using a graphical query editor.

6.4 UoL specification generator

The UoL Specification Generator (USG) module is used in the specification step of the requirements analysis life cycle. It aims to automatically generate the course requirements from the user requirements elicited in step 1. In fact a first draft of course requirements is generated using the template of course requirements presented is Section 5.

For functional course requirements, the short description of a learning activity is extracted from the instructional metadata of ADM and inserted to the following template:

FR_i: Learners in this UoL should achieve the following learning activity: *short description to insert from ADM*

For non-functional course requirements information is extracted from the four metadata categories

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT uol (act*)>
<!ATTLIST uol   uol_id ID #REQUIRED uol_name CDATA #REQUIRED>
<!ELEMENT act (inst, qual?, comp?, rel?)>
<!ATTLIST act   act_id ID #REQUIRED act_name CDATA #REQUIRED>
<!ELEMENT inst (short_des,act_cat?,act_type?,act_gran?,obj?,precond?,postcond?,inter_level?,diff?)>
<!ELEMENT short_des (#PCDATA)>
<!ELEMENT act_cat (#PCDATA)>
<!ELEMENT act_type (#PCDATA)>
<!ELEMENT act_gran (#PCDATA)>
<!ELEMENT obj (#PCDATA)>
<!ELEMENT precond (#PCDATA)>
<!ELEMENT postcond (#PCDATA)>
<!ELEMENT inter_level (#PCDATA)>
<!ELEMENT diff (#PCDATA)>
<!ELEMENT qual (usab, reliab, perf)>
<!ELEMENT usab (#PCDATA)>
<!ELEMENT reliab (#PCDATA)>
<!ELEMENT perf (#PCDATA)>
<!ELEMENT comp (LO, actor, service)>
<!ELEMENT LO (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
<!ELEMENT service (#PCDATA)>
<!ELEMENT rel (prereq, is_a_part, is_an_alter)>
<!ELEMENT prereq (#PCDATA)>
<!ELEMENT is_a_part (#PCDATA)>
<!ELEMENT is_an_alter (#PCDATA)>
```

Fig. 5. XML schema of ADM (ADM.dtd).

of ADM and inserted to the corresponding template:

For example for the actors requirements, the list of actors, which should be involved in a learning activity, is extracted from the components metadata of ADM, related to actors and inserted to the following template:

NFR$_j$: The activity in FR$_i$ should be involve the following actors: *List of actors to insert from ADM*

In fact, UoL description using ADM is represented in the repository with XML. It respects a predefined XML schema. Then the UoL specification is obtained through XSLT, which transforms the logical XML content of the UOL, into a formatted content in HTML.

Figure 5 shows the DTD (Document type Definition) of the ADM model. This DTD is written in a formal syntax that explains precisely which elements may appear where in the document and what the elements' contents and attributes are. It helps to specify what ADM does and doesn't allow. A validating parser will compare an XML instance of ADM to its DTD and lists any places where the document differs from the constraints specified in the DTD.

Figure 6 shows the XSLT (Extensible Stylesheet Language Transformations) that specifies the rules by which one valid XML instance of ADM is transformed into an HTML document readable on the web-browser. It contains a set of template rules. Each template rule has a pattern and a template. An XSLT processor compares the elements and other nodes in the input XML instance of ADM to the template-rule patterns in the stylesheet. The following figure is limited to functional requirements.

7. Case study

The following case study is used to illustrate how instructor' needs are expressed using the framework and how the needs are then represented in ADM. Systems Analysis and Design (IS240) is a core course offered to students in the second level at the department of information systems. The course includes 12 UoLs which are completed in one semester. We will focus on the fourth UoL which is about Domain Modeling [24]. In UoL4, students learn how to read, interpret, and create a domain model class diagram. The instructor who usually teaches this course organizes UoL4 into a set of activities including some collaborative activities. These activities are listed in Table 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
<xsl:template match="uol"><html><head><title>UoL specification</title></head>
<body><h1>UoL #: <xsl:value-of select="@uol_id"/></h1><h1>UoL name: <xsl:value-of select="@uol_name"/></h1><h2>Functional
requirements</h2><ol><xsl:apply-templates/></ol></body></html></xsl:template>
<xsl:template match="act"><li>Learners in this UoL should achieve the following learning activity: <xsl:value-of
select="inst/short_des"/></li>  </xsl:template>
</xsl:stylesheet>
```

Fig. 6. XSLT for generating UoL specification (Functional requirements)

Table 1. UoL description (User requirements) using ADM

Course : Systems Analysis and Design
Unit of Learning: Domain Modeling
ID: IS240-1400301

Activitiy	**Instructional metadata**	**Quality metadata**	**Components metadata**	**Relationship metadata**
1. Identifying classes using brainstorming	• Short description: use a checklist of all of the usual types of things typically found and perform brainstorming to identify domain classes of each type • Activity category: collaborative • Activity type: learning • Activity granularity: 0 • Objectives: Identify domain classes needed in the system • Precondition: students know how to identify use cases to define functional requirements • Post condition: students are able to use the brainstorming technique to identify classes. • Interactivity Level: High • Difficulty: Difficult	Reliability: • Steps of the brainstorming technique should be respected • Identified classes should be meaningful Performance: • Identifying classes from a simple case study (1 page) should not exceed 45 mn.	Resources: • Brainstorming technique definition • brainstorming technique steps • brainstorming technique application sample Actors/roles: • Groups of learners • Users from industry • Tutors Tools/services: • Checklist on classification of things	
2. Identifying classes using nouns	• Short description: Identify all of the nouns that come up when the system is described and determine if each is a domain class or an attribute • Activity category: individual • Activity type: learning • Activity granularity: 0 • Objectives: Identify domain classes needed in the system • Precondition: students know how to identify use cases to define functional requirements • Post condition: students are able to use the noun technique to identify classes. • Interactivity Level: Medium • Difficulty: Easy	Reliability: • Steps of the noun technique should be respected • Identified classes should be meaningful Performance: • Identifying classes from a simple case study (1 page) should not exceed 30 mn.	Resources: • Noun technique definition • Noun technique steps • Noun technique application sample Actors/roles: • Learner • Tutor Tools/services: • Wordnet	• Activity 2 is an alternative for Activity 1
3. Representing classes	• Activity category: individual • Activity type: learning • Activity granularity: 0 • Objectives: Represent a class in UML • Precondition: students know how to identify a class • Post condition: students are able to represent a class in UML. • Interactivity Level: Medium • Difficulty: Easy	Reliability: • Classes represented in UML should be correct	Resources: • Rules of class representation in UML • Samples of classes Actors/roles: • Learner Tools/services: • Power Designer tool	• Activity 1 is a prerequisite for Activity 3. • Activity 2 is a prerequisite for Activity 3.
4. Representing associations	• Short description: Identify all of the nouns that come up when the system is described and determine if each is a domain class or an attribute	•	•	• Activity 3 is a prerequisite for Activity 4.
5. Representing inheritance	• Short description: Identify all of the nouns that come up when the system is described and determine if each is a domain class or an attribute	•	•	• Activity 3 is a prerequisite for Activity 5.
6. Representing aggregation and composition	• Short description: Identify all of the nouns that come up when the system is described and determine if each is a domain class or an attribute	•	•	• Activity 3 is a prerequisite for Activity 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uol SYSTEM "adm.dtd">
<?xml-stylesheet type="application/xml"  system "uol.xsl"?>
<uol uol_id="IS240-1400301" uol_name="Domain modeling">
<act act_id="A1" act_name="Identifying classes using brainstrom"><inst><short_des> use a checklist of all of the usual types of things
typically found and perform brainstorming to identify domain classes of each type </short_des> <act_cat>collaborative</act_cat>
<act_type>learning</act_type><act_gran>0</act_gran><obj>Identify domain classes needed in the system</obj><precond>students
know how to identify use cases to define functional requirements</precond><postcond>students are able to use the brainstorming
technique to identify classes</postcond><inter_level>High</inter_level><diff>Difficult</diff></inst></act>
<act act_id="A2" act_name="Identifying classes using nouns"><inst><short_des>Identify all of the nouns that come up when the system
is described and determine if each is a domain class or an attribute </short_des></inst></act>
<act act_id="A3" act_name="Representing classes"><inst><short_des>Representing classes using UML grahical
symbols</short_des></inst></act>
<act act_id="A4" act_name="Representing associations"><inst><short_des>Representing associations using UML grahical
symbols</short_des></inst></act>
<act act_id="A5" act_name="Representing inheritance"><inst><short_des>Representing generalization - specialization using UML
grahical symbols</short_des></inst></act>
<act act_id="A6" act_name="Representing aggregation and composition"><inst><short_des>Representing aggregation and
composition using UML grahical symbols</short_des></inst></act>
</uol>
```

Fig. 7. UoL 4 description using ADM (XML instance).

7.1 Elicitation

In order to illustrate the instructor's work in describing a UoL we focus on eliciting a particular activity in UoL4 that is activity 3 "Representing classes". This activity consists in representing a class in UML. Representing classes is very helpful for students in modeling. This activity clarifies the concept of class and helps students mastering its graphical representation using UML. For this activity, the instructor did not create a description from scratch. Instead, he has chosen to reuse a similar activity done previously in a course about software engineering (IS324). Hence, the activity description is retrieved, is adapted and plugged into the working space as part of UoL4. The instructor has changed the name of this activity and specified the prerequisite constraint: that is to be sequenced after activities 1 and 2.

7.2 XML representation

Figure 7 shows the XML file that represents the ADM instance related to the Domain Modeling UoL. It is valid regarding the ADM DTD that it includes. Everything in this instance must match a declaration in the DTD

Figure 8 shows an XSL output of the UoL4 specification, which is generated by application of the XSLT (Fig. 6) on the XML instance of the UoL 4 description (Fig. 7). It is a set of six functional requirements corresponding to the six activities of UoL4 described in the case study.

7.3 Discussion

Expressing the instructors' needs using the proposed framework revealed many interesting points. First, focusing at an early stage of the course development exclusively on expressing the needs decouples the analysis from the course design and frees the instructor from thinking about the technology requirements and the available tools. Second, while observing the instructor at work in this analysis phase, it was clear that working with an interactive framework with the possibility to add and remove activities and to refine them through a set of cycles has led to the creation of activities that are very well articulated in the UoL. Third, the existence of activity instances has eased and expedited the instructor's task by just reusing existing activities which can be adjusted and plugged in the UoL. Last, the generator of the learning scenario is

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head><title>UoL specification</title></head>
<body><h1>UoL #: IS240-1400301</h1><h1>UoL name: Domain modeling</h1><h2>Functional requirements</h2><ol>
<li>Learners in this UoL should achieve the following learning activity:  use a checklist of all of the usual types of things typically found
and perform brainstorming to identify domain classes of each type</li><li>Learners in this UoL should achieve the following learning
activity: Identify all of the nouns that come up when the system is described and determine if each is a domain class or an attribute </li>
<li>Learners in this UoL should achieve the following learning activity: Representing classes using UML grahical symbols</li>
<li>Learners in this UoL should achieve the following learning activity: Representing associations using UML grahical symbols</li>
<li>Learners in this UoL should achieve the following learning activity: Representing generalization - specialization using UML grahical
symbols</li>
<li>Learners in this UoL should achieve the following learning activity: Representing  aggregation and composition using UML grahical
symbols</li></ol></body>
</html>
```

Fig. 8. XSL output of course UoL 4 specification (functional requirements).

a tool that helped the instructor reorganizing and testing the learning scenario sequence that best fit the students' background and optimizing the time allocated to each activity.

8. Conclusion

The software engineering framework proposed in this paper aims at supporting instructors during the analysis phase of e-course development. Instructors are solicited to provide their expertise as learning activities through a convenient graphical user interface. Hence the focus of the instructor is to provide a description of UoLs related to the e-course which are then converted automatically into specifications for further design and implementation.

Current work is focusing on enhancing the graphical user interface to provide more interactivity for the user and to allow reusing of UoLs the same way as activities are reused. Future work will emphasize on coupling the framework with the IMSLD in order to automatically generate a first design draft of UoLs. The specification generated for an e-course that complies with the DTD can be mapped to feed most of the categories of the IMSLD resulting in the complete e-course design. Also we are investigating the use of domain and learning design ontologies in the elicitation phase in an effort to standardize the UoL description, to enforce interoperability, and allow them to be shared and reused by communities of instructors.

Acknowledgements—This work was supported by the Research Center of the College of Computer and Information Sciences, King Saud University, Project # RC120913. The authors are grateful for this support.

References

1. L. A. Mills, G. Knezek and F. Khaddage, Information seeking, information sharing, and going mobile: three bridges to informal learning, *Computers in Human Behavior*, **32**, 2014, pp. 324–334.
2. M. Wang, Integrating organizational, social, and individual perspectives in Web 2.0-based workplace e-learning, *Information Systems Frontiers*, **13**(2), 2011, pp. 191–205.
3. A. Garcia-Cabot, E. Garcia-Lopez, L. de Marcos, L. Fernandez and J-M Gutierrez-Martnez, Adapting Learning Content to User Competences, Context and Mobile Device using a Multi-Agent System: Case Studies, *The International Journal of Engineering Education*, **30**(4), 2014, pp. 937–949.
4. D. Sampson, J. M. Spector, I. Aedo-Cuevas and N. S. Chen, Editorial on current advances in learning technologies, *educational technology & society journal*, **15**(4), 2012.
5. I. Sommerville, *Software engineering*, Pearson Education, 2011.
6. B. A. Nuseibeh and S. M. Easterbrook, Requirements engineering: a roadmap, in A. C. W. Finkelstein (ed), *The future of Software Engineering*, Companion volume to the proceedings of the 22nd International Conference on Software Engineering, ICSE'00, IEEE Computer Society Press, 2000.
7. What Is Instructional Design?, http://www-personal.umich.edu/~jmargeru/prototyping/design_definition.html, Accessed on 10/2/2015.
8. IMSLD (2003). IMS Learning Design. Information Model, Best Practice and Implementation Guide, Binding document, Schemas. http://www.imsglobal.org/learningdesign/index.cfm, Accessed on 20/2/2015.
9. H. Jonassen, Learning as activity, *Educational Technology*, **42**(2), 2002, pp. 45–51.
10. S. S. Liaw, H. M. Huang and G. D. Chen, An activity-theoretical approach to investigate learners' factors toward e-learning systems, *Computers in Human Behavior*, **23**, 2007, pp. 1906–1920.
11. P. Godfrey, R. D. Crick and S. Huang, Systems Thinking, Systems Design and Learning Power in Engineering Education, *The International Journal of Engineering Education*, **30**(1), 2014, pp. 112–127.
12. D. G. Sampson and P. Karampiperis, Towards next generation activity-based learning systems, *International Journal on E-Learning*, **5**(1), 2006, pp. 129–149.
13. LAMS Website, Learning Activity Management System, http://lamsfoundation.org/index.htm, Accessed on 25/2/2015.
14. J. Dalziel, Implementing learning design: the learning activity management system (LAMS), *Proceedings of the 20th annual conference of the Australasian society for computers in learning in tertiary education (ASCILITE)*, Adelaide, Australia, 7–10 December, 2003, pp. 593–596.
15. R. Koper and B. Olivier, Representing the learning design of units of learning, *Educational Technology & Society*, **7**(3), 2004, pp. 97–111.
16. E. Cloete, A. V. D. Merwe and L. Pretorius, A process modeling approach to requirements elicitation to incorporate e-learning as a core learning strategy, *Proceeding of the seventh world conference on integrated design and process technology*, Austin, Texas, 2003.
17. A. G. Sutcliffe, N.A.M. Maiden, S. Minocha and D. Manuel, Supporting scenario-based requirements engineering, *IEEE transactions on software engineering*, **24**(12), 1998, pp. 1072–1088.
18. M. K. Zarinah and S. S. Siti, Supporting collaborative requirements elicitation using focus group discussion technique, *International Journal of Software Engineering and Its Applications*, **3**(3), 2009, pp. 59–70.
19. R. Koper and C. Tattersall, *Learning design—A handbook on modeling and delivering networked education and training*. Springer-Verlag, Berlin, Heidelberg, 2005.
20. R. Koper and Y. Miao, Using the IMS LD standard to describe learning designs, in L. Lockyer, S. Bennet, S. Agostinho and B. Harper (eds), *Handbook of research on learning design and learning objects issues, applications and technologies*, IGI Global, Hersey, 2008, pp. 41–86.
21. A. Kleppe, J. Warmer and W. Bast, *MDA explained. The model driven architecture: practice and promise*, Addison Wesley, 2003.
22. Object Management Group—Model Driven Architecture (MDA) MDA Guide rev. 2.0 OMG Document ormsc/2014-06-01, http://www.omg.org/cgi-bin/doc?ormsc/14-06-01, Accessed on 12/2/2015.
23. C. Martel, L. Vignollet, C. Ferraris, J. P. David and A. Lejeune, LDL: an EML alternative, *Proceedings of the Sixth International Conference on Advanced Learning Technologies—ICALT*, Kerkrade, The Netherlands, 2006.
24. J. W. Satzinger, R. B. Jackson and S. D. Burd, *Introduction to systems analysis and design, 6th edition*, Cengage learning, 2012.

Azeddine Chikh is faculty member at King Saud University—Saudi Arabia. Prior to joining KSU, he worked as Chair of the Computer Science Department at University of Tlemcen, Algeria. He holds a PhD in Information Systems from National Institute of Computer Sciences in Algeria and Paul Sabatier University in France in 2004, a Masters in Information Systems from National Institute of Computer Sciences in Algeria 1994, a Masters in e-learning from Switzerland in 2007, and a graduate certificate in systems engineering from the University of Missouri Rolla, USA, in 2003. He worked as a researcher at the European project « PALETTE: Pedagogically sustained Adaptive LEarning Through the exploitation of Tacit and Explicit knowledge ». His research interests include learning design; communities of practice and social networks; document reuse; ontologies and semantic web; and software requirements engineering.

Jawad Berri is faculty member at King Saud University— Saudi Arabia. He received his Ph.D. in Computer Science from Paris-Sorbonne University in France in 1996. Jawad's research interests focus on context-aware web systems, learning technologies and natural language processing. He has been involved in many projects related to mobile learning, semantic web, automatic summarization, web information filtering and mobile agents for web information discovery. He worked as a researcher at the CNRS – the French National Research Center, the Computer Science Institute at the University of Zurich – Switzerland and Sonatrach – the Algerian Petroleum and Gas Corporation. Jawad's contributions in research projects in the industry and academia led to the publication of papers in numerous journals and conferences. Jawad is a senior member of the IEEE.