Customer Satisfaction through E-Learning Software Product Line

Amina Guendouz
CS Department
Saad Dahlab University
Blida, Algeria
Email: guendouz.amina@yahoo.fr

Djamal Bennouar
CS Department
Akli Mohand OulHadj University
Bouira, Algeria
Email: dbennouar@gmail.com

Abdelouahab Ramdani, Hamza Mazari
CS Department, Saad Dahlab University
Blida, Algeria
Email: wahab.inf@gmail.com,
hamza.ntj@hotmail.com

***Abstract*— As online education becomes a basic need for several organizations, a variety of Learning Management Systems is proposed on the market. However, available systems do not satisfy all the needs of different institutions, which push them to develop their own systems. Since developing and maintaining new software are cost, time and effort consuming, and with the increasing demand on e-Learning systems, it becomes necessary to find an efficient solution that allows the fast development of systems and overcomes the before-mentioned issues. We strongly believe that adopting a software product line approach in e-Learning domain can bring important benefits. In this paper, we present the development process of an e-Learning software product line. Throughout the development process, we demonstrate how this approach allows us to satisfy the variable needs of customers and benefit from the systematic large scale reuse at the same time.**

Keywords–E-Learning; Software Product Line; reusability; variability management.

I. INTRODUCTION

Nowadays, the Internet knows a spread use in several fields, including the education. Taking advantage from the benefit of using the Internet, organizations seek to provide an efficient and less expensive way of education in terms of time, cost and effort. Remote training, virtual learning, or electronic learning (e-Learning) means the use of Information and Communication Technologies (ICT) in education to improve the process of teaching-learning.

In order to satisfy the needs of the different institutions, various e-Learning applications have been proposed, the most known are Learning Management Systems (LMSs). A LMS, also known as Virtual Learning Environments (VLE), is the infrastructure that delivers and manages instructional content, identifies and assesses individual and organizational learning or training goals, tracks the progress towards meeting those goals, and collects and presents data for supervising the learning process of organization as a whole [12].

In spite of their important advantages, LMSs present several limitations. Dalsgaard [3] argues that LMSs are limited to cover only administrative issues, and suggests the necessity to go beyond LMSs in e-Learning to improve interactions between students and instructors. On the other hand, a survey on LMSs that has been carried out for 113 European institutions [9] revealed that a large number of the LMS systems used in Europe are commercial systems developed locally, or self-developed systems built by the institutions. Only a few commercial systems are used by several institutions, which means that institutions tend to create their own e-Learning systems to fulfill their specific requirements. García-Peñalvo et al. [19] announced that, despite the high levels of LMS adoption, these systems have not produced the expected learning outcomes yet. They mentioned that among the main shortcomings of LMSs the failure to take into account the user. Other recent studies [20][21][22] show that LMSs do not satisfy all the needs of teachers and students which push them to use social networks, cloud based services and mobile applications in order to complement the lack of LMSs, and suggest that students need learning environments which are better adapted to their needs.

In order to overcome these issues, we suggest the use of Software Product Line (SPL) approach for the development of e-Learning applications. E-Learning applications could be implemented in a variety of settings: for schools and universities to compliment or enhance classroom learning, for corporations to provide training and certification for their employees, and for organizations to provide e-learning courses to a larger learners population virtually anywhere in the world.

However, all of these applications share a set of common software elements and differ by some variable parts. So, the adoption of a SPL approach in the e-Learning domain seems to be a promising solution. On one side, to overcome the limitations of LMS systems, and on the other side, to provide institutions with e-Learning applications that fit their own requirements. Furthermore, SPL Engineering (SPLE) aims to share the development work of a set of product using common means of production, in order to reduce the costs and effort of development, maintenance and test, decrease time to market and improve quality.

In this paper, we show how to build a SPL for e-Learning applications. The remainder of this paper is organized as follows: Section 2 introduces SPL approach and presents the SPLE process that we will follow after that to develop our e-Learning SPL. Section 3 shows the different steps of the development of an e-Learning product line, mainly domain engineering. Section 4 comments on related work, while Section 5 summarizes the paper and outlines future work.

II. SOFTWARE PRODUCT LINE ENGINEERING

A SPL is "A set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" [6]. SPL approach aims to systematize the reuse throughout all the software development process: from requirements engineering to the final code and test plans. The purpose is to reduce the time and cost of production and to increase the software

 ISBN: 978-1-61208-361-2

quality by reusing elements (core assets) which have been already tested and secured. These objectives can be realized by putting in common development artefacts such as requirement documents, design diagrams, architectures, codes (reusable components), procedures of test and maintenance, etc.

SPL approach aims to improve reuse while maintaining diversity between products. This could be done by "Variability management". Variability management is a key activity that usually affects the degree to which a SPL is successful [2]. Variability refers to the ability of an artefact to be configured, customized, extended, or changed for use in a specific context [1]. This variability must be defined, represented, exploited, implemented, evolved, etc. – in one word managed – throughout SPL engineering [11].

SPL Engineering (SPL) relies on a fundamental distinction between two activities [5][11]: SPL development and software production. SPL development aims to develop and maintain the base of reusable elements while software production aims to produce final applications according to customer's needs. As mentioned in Section 1, the main shortcoming of LMSs is the production of generic applications that do not meet the specific requirements of customers. Adopting a SPL process for the development of e-Learning applications will permit customers to be involved in the development process of their applications which gives them the opportunity to customize applications according to their specific needs. Moreover, e-Learning application developers will benefit from the large scale reuse and, thus, the reduction of time, effort and cost of development.

Based on SPL approach, we propose, as shown in Figure 1, a development process for e-Learning SPL. It is composed of two sub-processes: Domain engineering and Application engineering. Domain engineering (correspond to SPL development activity) includes domain analysis, domain design and domain implementation activities. The purpose of domain engineering is to produce reusable core assets and to provide the effective means that help in using these core assets to build a new product within a product line. A core asset is a reusable artifact or resource that is used in the production of more than one product in a SPL. A core asset may be an architecture, a software component, a domain model, a requirements statement or specification, a document, a plan, a test case, a process

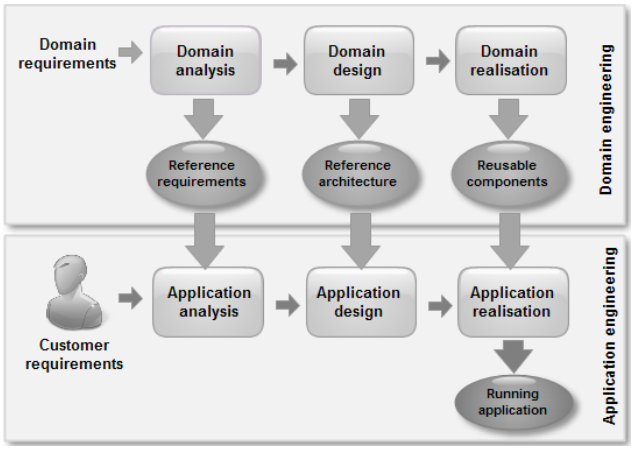

Figure 1. Software product line engineering process.

description, or any other useful element of a software production process [6]. The main outputs of this process are: reference requirements, reference architecture and reusable components.

Application engineering (corresponding to software production activity) consists in developing the final products, using the core assets and the specific requirements expressed by customers. This process is similar to traditional development process; however, each step is facilitated by the reuse of the outputs of the first process. The result of this process is an application ready to be used.

III. E-LEARNING SOFTWARE PRODUCT LINE ENGINEERING

In this section, we show the development process of our e-Learning product line focusing on the first sub-process: domain engineering.

A. Domain Engineering

As a preliminary activity of domain engineering, the scope of the SPL must be defined. In our case, e-Learning product line intends to cover the e-learning applications used by schools and universities providing online courses to their students, companies which provide online training to their employees and organizations that supply online courses to learners anywhere in the world. Domain engineering consists of three activities that are domain analysis, domain design and domain realization (Figure 1).

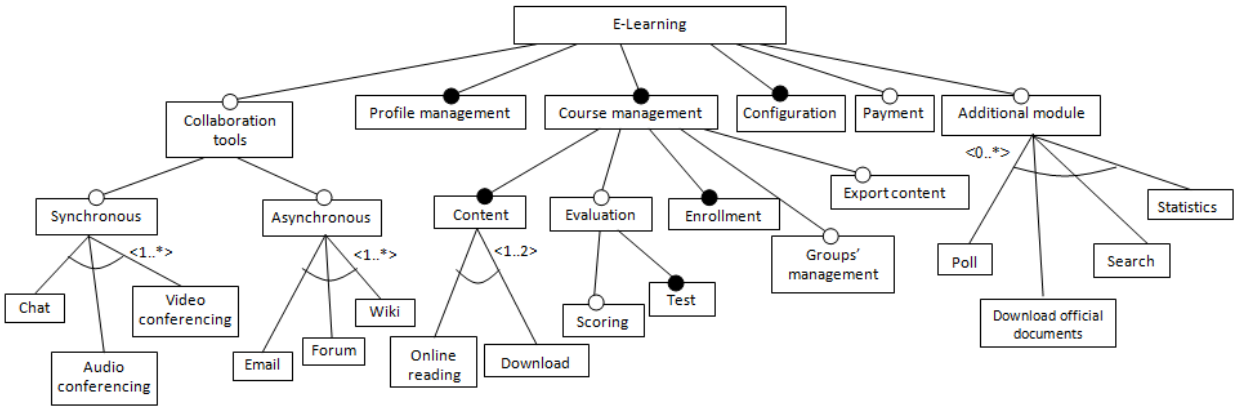

Figure 2. Capability feature diagram for e-Learning product line.

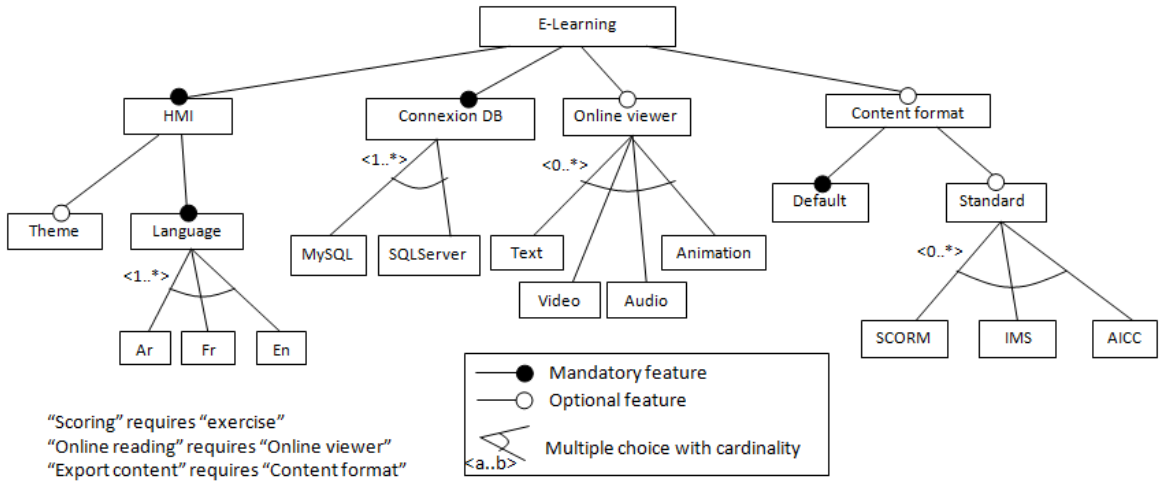

Figure 3. Implementation feature diagram for e-Learning product line.

1) Domain Analysis: The goal of domain analysis is to extract and document the similarities and variations between the SPL members. To document the common and variable features of our product line, we have used the Feature Model. The Feature Model is the first language dedicated to the modeling of variability; it was first introduced in the Feature-Oriented Domain Analysis (FODA) method [4]. It has known a broad use in the field of SPLE and several extensions [13][14][15], since it is a simple and easy to use language in comparison with other more complex modeling languages such as: Unified Modeling Language (UML) [23] and Business Process Modeling Notation (BPMN) [24]. The feature model is generally described by a hierarchy of the set of features of a system or what is called feature tree [5]. Figures 2 and 3 show a part of the feature model of our case. For the notation, we must note that in the case of multiple choices we have used only cardinality notation to avoid cluttering the diagram with different notations, for us cardinality is sufficient to represent all kinds of choices.

The feature model we constructed is divided into two diagrams according to the type of features it includes. Features in the first diagram called capability features, (Figure 2), represent the functionalities provided by the system. This diagram shows that the main features of an e-Learning application are "Profile management" and "Course management" and "Configuration". However, the application may include other functionalities such as:

- Online payment in the case of paid courses provided for example by private organizations.
- Interaction with learners through "Collaboration tools". Collaboration tools may be synchronous (chat, video conferencing, etc.) or asynchronous (e-mail, forum, wiki, etc.).
- Additional modules such as: statistics, search, download official documents (bulletin, attestation, etc.) and others. The cardinality 0..* for the feature "Additional module" means the possibility of adding new sub-features and so the possibility to extend the product line to cover new requirements.

A "Course management" must contain at least "Content" and "Enrollment" features, but it can include other optional features according to the usage context, such as: "groups' management", "Export content", and "Evaluation". The evaluation (if selected) may include several types of questions, for instance: in some cases text questions are sufficient, in other cases diagrams or audio recordings are needed. We do not show the whole feature model for the sake of brevity and space reasons.

The second diagram reported in Figure 3 represents the implementation features of the system; it means implementation details at lower and more technical levels. An e-Learning application must connect to a data base and supply a Human Machine Interface (HMI). If the application provide "Online reading" of the course's content, this require an "Online viewer" which differ according to the type of the content (text, video, sound or animation). The content of a course can be exported in several formats: default format provided by the system or other standard formats such as: Sharable Content Object Reference Model (SCORM) [26], IMS Global Learning Consortium (IMS GLC) [27] or Aviation Industry Computer-Based Training Committee (AICC) [28]. Constraints at the bottom of the diagram are used to express dependencies between features in the same diagram or between capability and implementation features.

2) Domain Design: The purpose of the domain design is to establish the generic software architecture of the product line. Variability identified during domain analysis must be explicitly specified in the product line architecture.

In our case, we have chosen Orthogonal Variability Model (OVM) [5] to represent variability in the design model. OVM consist of a set of Variation Points (VP) and Variants (V). OVM is based on a separation between variability model and other artefacts in order to decrease models complexity [16]. A variation point or variability subject shows an aspect of variability within the product line. Variants or variability objects are the different shapes of a variability subject. Using OVM allows us to represent variability in the architecture view without having to extend the design language. Variability is modeled separately from the component diagrams and related to this latter by means of traceability links.

 ISBN: 978-1-61208-361-2

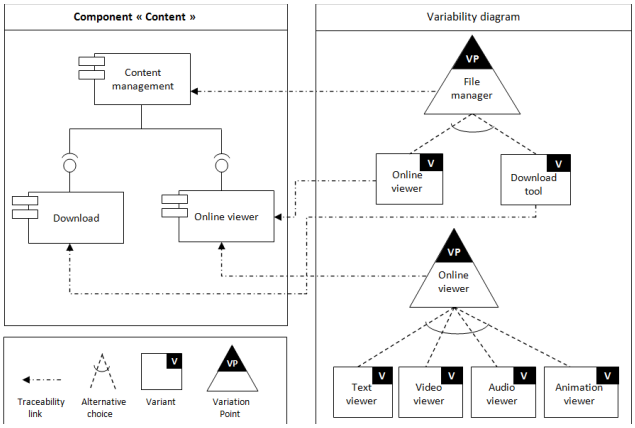

Figure 4. Variability model for "content" component.

To create our OVM, we relied on the Feature Model constructed in domain analysis phase. Figure 4 shows the variability modeling of the component "content" from the set of components of e-Learning product line. In this example, the component "Online viewer" will be implemented in one or more versions: text viewer, video viewer, audio viewer, or animation viewer.

3) Domain Realization: The main object of this step is to create a set of reusable software components. The components that have been identified in the previous step are detailed, planned and implemented to be reused in different contexts. To build our components we have used Java Enterprise Edition (J2EE) [25]. The result of this step is not a running application, but rather, a set of configurable and loosely coupled reusable components, that will be assembled during application derivation step.

B. Application Engineering

One can distinguish between two kinds of variability [18]: (i) product line variability which is specific to SPLE and describes the variation between the members of a SPL, and (ii) software variability that refers to the ability of a software element to be changed or customized for use in a particular context. Product line variability is resolved (bound) during Application Engineering through several binding times (design, implementation, compilation, assembly) [17], while software variability is bound after the delivery of an application (configuration and runtime). In order to ensure an efficient support of customer's needs, customers are involved not only at runtime but also at application derivation (Application Engineering) as well as application configuration steps (Figure 5).

During application engineering, e-Learning applications are derived using the core assets and customer's specific requirements. The feature model that we have defined represents the decision space for our SPL. For each new application, we select the relevant features from the feature diagram according to the specific customer requirements. The feature model of the particular application is then used to specify which variants must exist in the architecture model. As a result, we obtain an architecture model without variability, and which includes only the components of the derived application, in addition to components that implement the specific requirement if they exist. Finally, according to the application's architecture model, we select the components from the base of reusable components obtained in domain realization. In the case where the particular application needs components that have not been predicted in Domain engineering, these application-specific components must be implemented and then assembled with the selected reusable components to create the final running application.

During application configuration, customer can decide about variation points that have been delayed to this binding time, for instance, he can specify the language, the website address and name, the database driver, the administrator profile, etc. Other variations such as: courses organization, additional modules, theme selection, access rights of users, etc., might be delayed until runtime. Allowing customers to customize their applications through several steps (derivation, configuration and runtime) lead to more flexible applications and thus better user satisfaction.

As mentioned in section III, customers could be teachers in schools and universities and institutions providing free or paid online courses. When delivering the final applications, they will have neither extra-functionalities that they does not need, nor lacking functionalities that they must integrate themselves as in the case of LMSs. So, the customer's satisfaction is assessed according to the conformance of the provided functionalities to the required ones.

IV. Related Work

SPL was first used in e-Learning domain to develop and reuse digital educational content [7][8]. Pankratius et al. proposed the Product Lines for Digital Information Products (PLANT) approach to deal, in a general way, with the issues encountered in content reuse for e-Learning platforms. In this case, the reusable elements are a mixture of content and software, since online courses may contain, more than texts, programs and animations.

Another work using SPL approach to develop an auxiliary e-Learning application is presented by Sanchez et al. [10]. They use SPL engineering to develop e-Learning Web-miner product line, a family of data-mining applications aiming to assist educators involved in virtual education by extracting and providing useful information that these educators can use to improve the learning-teaching process.

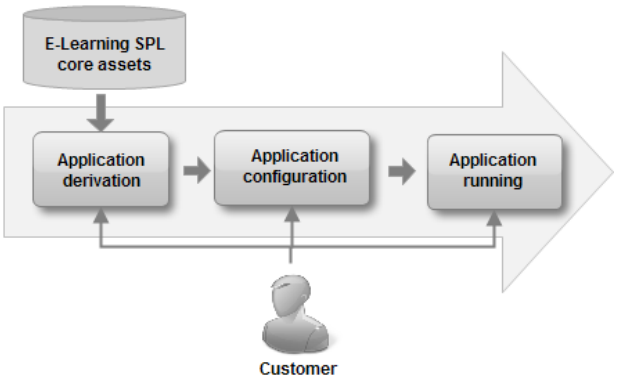

Figure 5. Application customization.

 ISBN: 978-1-61208-361-2

To the best of our knowledge, the use of SPL principles in the domain of e-Learning was limited to reuse online courses [7][8], or to develop data-mining applications related to e-Learning platforms [10]. But, there is no other SPL for e-Learning applications similar to the one we presented in this paper. By the present work, we show that e-Learning is a wide domain that includes several applications, characterized by an important set of common features and vary in some aspect, the adoption of SPL approach in this field can obviously bring important benefits not only to developers but also to users.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed the use of SPLE approach to develop e-Learning applications. The presented work aims to overcome the shortcomings of LMSs, mainly by satisfying the variable requirements of customers, providing more flexible applications, and to benefit from the advantages of SPL engineering. Using SPL approach in such a broad field allows developers to reduce costs and effort of both production and maintenance, to decrease significantly time of development and to improve quality.

The paper presented the different steps of the development process of an e-Learning product line, focusing on domain engineering. This latter result in a set of core assets: the domain requirements documented by the feature model, the reference architecture models including variability presented by OVM model and the software components. This base of core assets will be reused to simplify the development of each new member of e-Learning SPL during application engineering. Moreover, customers are prompted to express their needs throughout the application instantiation steps in order to reach better satisfaction.

As future work, we intend to improve our e-Learning product line by decomposing it into a set of sub-SPLs, each one intended for an e-Learning subfield (primary, secondary, university, paid courses, etc.). This will allow us to cover a broader scope while ensuring efficient variability management. It will be also important to define an automatic method of derivation to improve the application engineering process.

REFERENCES

[1] F. Bachmann and P. Clements, “Variability in software product lines,” technical report CMU/SEI, 2005.

[2] L. Chen, M. Babar, and A. Nour, “Variability management in software product lines: a systematic review,” in SPLC, San Francisco, California, 2009, pp. 81-90.

[3] C. Dalsgaardy, Social software: e-learning beyond learning management systems. European Journal of Open, Distance and E-Learning, 2006, no. 2, [Online]. Available from: http://www.eurodl.org/index.php?article=228 2014.6.6

[4] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson, “Feature oriented domain analysis (FODA) feasibility study,” Technical Report CMU/SEI-90-TR-21, 1990.

[5] P. Klaus, G. Bockle, and F. van der Linden, Software product line engineering: foundations, principles, and techniques. Springer, 2005.

[6] L. Northrop and C. Clements, “A framework for software product line practice,” Version 5.0. [Online]. Available from: http://www.sei.cmu.edu/ 2014.6.6

[7] V. Pankratius, Product Lines for Digital Information Products. Information Systems, 2007.

[8] V. Pankratius and S. Wolried, “A strategy for content reusability with product lines derived from experience in online education,” in International Conference on Software Engineering (ICSE), USA, 2005, pp. 128-146.

[9] M. Paulsen, “Experiences with learning management systems in 113 european institutions,” Educational Technology and Society, vol. 6 (4), 2003, pp. 134-148.

[10] P. Sanchez, G. Diego, and Z. Marta, “Software product line engineering for e-learning applications: a case study,” in 2012 International Symposium on Computers in Education (SIIE 2012), Andorra, 2012, pp. 1-6.

[11] F. V. der Linden, and E. R. K. Schmid, Software product lines in action: the best industrial practice in product line engineering. Springer, 2007.

[12] R. Watson, “An argument for clarity: what are learning management systems, what are they not, and what should they become?” TechTrends, vol. 51, 2007, pp. 28–34.

[13] K. C. Kang, K. Sajoong, L. Jaejoon, K. Kijoo, J. Gerard, and S. Euiseob, “FORM: A feature-oriented reuse method with domain-specific reference architectures,” Annals of Software Engineering, vol. 5, 1998, pp. 143–168.

[14] M. L. Griss, F. John, and A. Massimo, “Integrating feature modeling with the RSEB,” in Proceedings of the Fifty International Conference on Software Reuse, Victoria, Canada, 1998, pp. 76-85.

[15] K. Czarnecki and C. H. P. Kim, “Cardinality-based feature modeling and constraints: a progress report,” in International Workshop on Software Factories at OOPSLA’05, San Diego, California, USA, 2005, pp. 16-20.

[16] K. Pohl and A. Metzger, “Variability management in software product line engineering,” in Proceedings of the 28th international conference on Software engineering, New York, NY, USA, 2006, pp. 1049-1050.

[17] R. Capilla and J. Bosch, “Binding time and evolution,” Systems and software variability management, 2013, pp. 57–73.

[18] A. Metzger, K. Pohl, P. Heymans, P. Schobbens, and G. Saval, “Disambiguating the documentation of variability in software product lines: a separation of concerns,” in 15th IEEE International In Requirements Engineering Conference, Delhi, 2007, pp. 243 - 253.

[19] F. J. García-Peñalvo, M. Á. Conde, M. Alier, and M. J. Casany, “Opening learning management systems to personal learning environments,” Journal of Universal Computer Science, 17(9), 2011, pp. 1222-1240.

[20] V. Stantchev, R. Colomo-Palacios, P. Soto-Acosta, and S. Misra, “Learning management systems and cloud file hosting services: A study on students’ acceptance,” Computers in Human Behavior, Vol. 31, February 2014, pp. 612–619.

[21] M. A. Conde, F. García, M. J. Rodríguez-Conde, M. Alier, and A. García-Holgado, "Perceived openness of Learning Management Systems by students and teachers in education and technology courses,"Computers in Human Behavior, Vol. 31, February 2014, pp. 517-526.

[22] Z. Du, X. Fu, C. Zhao, Q. Liu, and T. Liu, "Interactive and Collaborative E-Learning Platform with Integrated Social Software and Learning Management System,"Proceedings of the 2012 International Conference on Information Technology and Software Engineering, Lecture Notes in Electrical Engineering, Vol. 212, 2013, pp 11-18.

[23] T. Baar, A. Strohmeier, A. Moreira, and S. J. Mellor, UML 2004 - The Unified Modeling Language. Springer, 2004.

[24] J. Mendling, M. Weidlich,and M. Weske , Business Process Modeling Notation. Springer, 2011.

[25] E. Armstrong, J. Ball, S. Bodoff, D. B. Carson, I. Evans, D. Green, and E. Jendrock, The J2EE 1.4 tutorial. Sun Microsystems, 2004. Available from: http://docs.oracle.com/javaee/1.4/tutorial/doc/ 2014.7.4

[26] “Sharable Content Object Reference Model,” [Online]. Available from: http://scorm.com/scorm-explained/ 2014.7.4

[27] “IMS Global Learning Consortium,” [Online]. Available from: http://www.imsglobal.org/ 2014.7.4

[28] “Aviation Industry Computer-Based Training Committee,” [Online]. Available from: http://www.aicc.org/joomla/dev/ 2014.7.4