Towards the Establishment of a Software Product Line for Mobile Learning Applications

Venilton Falvo Júnior
ICMC/USP, Brazil
venilton@icmc.usp.br

Nemésio F. Duarte Filho
ICMC/USP, Brazil
nemesio@icmc.usp.br

Edson OliveiraJr
UEM/PR, Brazil
edson@din.uem.br

Ellen Francine Barbosa
ICMC/USP, Brazil
francine@icmc.usp.br

***Abstract*—The enormous popularity of mobile devices in the society has motivated the development of mobile learning applications. In spite of the benefits with regard to teaching and training, the existing learning applications still have to address issues and challenges related to the development, reuse and architectural standardization. On the other hand, researches have been carried out to employ the Software Product Lines (SPL) approach through the development of mobile learning applications. A SPL focuses on software reuse and has been successfully applied for specific domains. In this context, this paper proposes M-SPLearning, one SPL particularly established to the mobile learning applications domain. The main goal of M-SPLearning is to provide benefits with regard to the overall quality, domain comprehension, and reduction of the time spent in the development and maintenance of m-learning applications.**

Keywords- m-learning; software product line; mobile learning applications; proactive adoption model

INTRODUCTION

In recent years, learning environments have shown an increasing importance, playing a fundamental role in teaching and training activities, both in academic and industrial settings. Together with the advent of ubiquitous and mobile computing, learning environments have also contributed for a new modality of education – the m-learning (mobile learning) [1, 2]. As an attempt to analyze the main motivations for m-learning, O'Malley et al. [3] emphasize the impact of technological advances, such as intelligent interfaces, contextual modeling applications and the recent progresses in the wireless communications area, which altogether have provided several new and innovative perspectives for technology users.

M-learning introduces flexibility to the learning process, since the creation, exchange and access to information occur naturally due to the ubiquity of mobile devices. In this sense, apprentices are able to decide when, how and where they feel more comfortable to learn [3]. On the other hand, one of the main problems of m-learning is the lack of a well-defined standard with respect to the means of information access. Due to the large number of mobile devices available in the market, the production of content for these devices becomes strongly dependent of issues such as manufacturer and operating system, for instance [2].

In a different but related perspective, the introduction of object-oriented (OO) concepts, component-based development and service-oriented development have attracted the interest of the software community to the opportunities and benefits of code reuse. The success of such initiatives have stimulated the reuse in several stages of the software development process, including in artifacts such as documents, specifications and models, further increasing the perspective of cost reduction and return on investment (ROI) [4].

The evolution of these ideas has led to the concept of Software Product Line (SPL), which represents a paradigm change in the regard of the traditional software development. Instead of developing software "project-to-project", organizations should now focus their efforts on creating and maintaining a core asset, which would be the basis for the construction of specific products for a given domain [5].

Motivated by this scenario, in this paper we propose a SPL for the development of m-learning applications, named M-SPLearning. The main goal is to investigate the benefits of systematic reuse of an SPL in the context of m-learning. At the very end, the idea is to promote the overall quality, domain comprehension, and reduction of the time spent in the development/maintenance of m-learning applications.

This paper is organized as follows. In Section II, the background for our work is summarized. In Section III, we describe the main aspects of M-SPLearning through the process used for its creation. In Section IV, we discuss a preliminary evaluation of the proposed SPL. In Section V, we briefly discuss the threats to validity. Finally, in Section VI, we summarize our conclusions and perspectives for future work.

BACKGROUND

SPL represents a new paradigm in Software Engineering, providing expressive results in terms of cost, schedule and quality [6]. Linden et al. [5] describe a SPL as a set of software systems sharing common features that satisfy a specific need for a particular market segment, developed from core assets in a systematic way, usually formed by a software architecture and its components.

The concept of SPL is suitable to domains in which there is a demand for products that have common features but which also contain a well-defined set of variabilities. Despite its relevance, the required activities to adopt the SPL concept are not trivial, demanding considerable time and effort. Krueger [7, 8] presents three different adoption models to support the establishment of a SPL: (i) proactive; (ii) reactive; and (iii) extractive. The proactive model is fully supported by the scope of the required systems, being appropriate when the requirements for the set of products to be developed are stable and can be previously defined. Therefore, each activity has to be finished before the next one be started.

In the reactive model, the SPL incrementally evolves whenever there is a demand for new products or new requirements are specified for the existing products. This approach is suitable when it is not possible to predict the requirements for each specific product.

The extractive model reuses one or more existing software to the initial base of the SPL. To be an effective choice, this model should not require complex technologies for the development of the SPL and must allow the reuse of existing software without the need of a high level of reengineering.

Other relevant concept refers to m-learning. This new type of electronic learning takes place when the interaction among the actors of the learning process is performed through mobile devices (tablets, smartphones, PDAs, etc). As a new and emerging paradigm, there are several attempts for defining it. According to O'Malley et al. [3], m-learning refers to any kind of learning that occurs when the apprentice is not in a fixed place, or when one takes advantage of learning opportunities provided by mobile devices, thereby relating technological and mobility concepts. Ozdamli and Cavus [9] describe m-learning as an activity that allows individuals to be more productive when they consume, create or interact with information, supported by mobile devices.

No matter the definition adopted, the interaction with learning applications through mobile devices provides benefits that go beyond accessibility, convenience and communication [10]. However, in spite of the advantages offered and even with the increasing demand for m-learning applications, there are few works addressing development issues in this new learning setting. One of the researching initiatives in terms of the development of m-learning applications can be found in Duarte Filho and Barbosa's work [11]. In short, the authors proposed a requirements catalog for m-learning environments. The catalog was established from the results of a systematic review conducted in this domain. To facilitate the understanding and the maintenance of the catalog, a three-level hierarchical structure (criteria, requirements and description) was adopted. Additionally, based on the knowledge of domain specialists, the requirements were prioritized in order to reflect the main experiences and needs in the m-learning setting.

Duarte Filho and Barbosa [11] also suggest that the requirements defined in the catalog may serve as a basis for: (1) the specification of a quality model for m-learning environments; and (2) the establishment of a reference architecture for m-learning environments. Actually, quality standards can support the definition of requirements, thereby contributing to the establishment of a high quality architecture.

The SPL architecture (PLA) plays a central role to successfully generate specific products taking into account the development and evolution of a SPL. It represents, in an abstract level, the architecture of all potential products for a specific domain. The PLA addresses the SPL design decisions by means of similarities and variabilities [12]. Thus, the PLA evaluation can be seen as one of the most important activities throughout a SPL life cycle [5]. In this sense, the requirements catalog proposed by Duarte Filho and Barbosa [11] and the adoption models proposed by Krueger [7, 8] have been investigated as the basis for the M-SPLearning construction.

M-SPLEARNING: CONSTRUCTION PROCESS

Before starting to develop M-SPLearning, we looked for some related works dealing with the construction of SPLs (Table I). Although considering different domains, all the works analyzed provide relevant information regarding the techniques and approaches used in the construction and adoption of SPLs. For the sake of space, these works will not be detailed herein.

TABLE I. SUMMARY OF THIS AND RELATED WORK

Work	Domain	Adoption Model	Architectural Base
M-SPLearning	M-learning	Proactive	Component
Dalmon et al. [13]	Interactive modules	Extractive (5 applications)	Component
Marinho et al. [14, 15]	M-guides	Extractive (57 applications)	SOA
Pascual et al. [16]	Pervasive systems	Proactive	Component and Aspect

Linden et al. [5] claim that one of the most critical activities in creating a SPL refers to the scoping of the target domain. To define the initial scope of M-SPLearning, we mainly considered the requirements catalog proposed by Duarte Filho and Barbosa [11]. According to the authors, the catalog intends to reflect, in a high level basis, the experience gained from developers and researchers in this new modality of learning. Furthermore, the catalog is generic and embracing, benefiting its adoption for different purposes in the m-learning domain.

Still with regard to scoping issues, since mobile applications can be built using a native development approach, the m-learning domain encompasses several different operating systems. Thus, to define an acceptable domain in terms of scope, we had to select a single operating system. Our choice, Android OS, was based on the amount of devices that each operating system controls. According to Llamas et al. [17], Android OS owns 68.8% of the world's smartphones market, which represents nearly 500 million devices.

Having delimited the scope, the adoption models proposed by Krueger [7, 8] were analyzed to identify the most appropriate to the construction of M-SPLearning. The specificities and features of the mobile domain, along with the existence of a requirements catalog for m-learning environments [11], have motivated the choice of the proactive model. Such a proactive approach comprises four main phases, described next.

A. *Domain Analysis*

According to Krueger [7, 8], at this phase the domain is analyzed in order to identify the variation in the specific products from a SPL. In this sense, requirements catalog proposed by Duarte Filho and Barbosa [11] was analyzed with respect to the ISO/IEC 25010 – International Standard for Software Product Quality (successor of the ISO/IEC 9126 standard) [18]. The aim was to identify missing or disconnected requirements.

From the analysis conducted, we noticed that most of the requirements were equivalent to the features/sub-features established by the ISO/IEC 25010 standard. However, some requirements had to be rearranged and/or renamed and, in a fewer cases, added and/or removed.

The following step consisted of identifying the variabilities incorporated by the specific products of the SPL. Variability is one of the most important issues in designing a SPL, reflecting the diversity and commonality of its artifacts [19]. Therefore, the precise and explicit representation of variabilities makes it possible the generation of specific products in a SPL.

Variabilities may be identified and represented by the concept of features [20]. A feature is defined as a system characteristic that is relevant and visible to the end user [21]. Features are usually represented by a feature model, i.e., a hierarchical representation that captures the structural relationships among the features of a specific domain.

Feature models are visually represented by means of feature diagrams. Figure 1 illustrates the feature diagram representing the requirements catalog for M-SPLearning. The interpretation of the feature diagram is straightforward and its construction is in agreement with the concepts and guidelines proposed by Kang et al. [21]. Shortly, each requirement in the catalog was evaluated considering its relevance in the SPL domain and, if appropriate, mapped as feature. The primary features are summarized as follows:

- *Pedagogical:* incorporates educational and pedagogical requirements in order to facilitate and support teaching and training activities. This feature and its sub-features are represented as mandatory due to their relevance in the mobile learning domain. The only optional sub-feature is *interactivity*, which allows communication with social networks;

- *Usability:* addresses relevant issues with respect to the visual interface of a product, being crucial to the software market acceptance. This feature assures that all products generated by the SPL follow an usability standard, adding quality to the final product. Since usability is a global feature of the product and not a specific feature, it is defined as mandatory and abstract.

- *Compatibility:* encompasses coexistence and the ability of a product to exchange information with other systems in the same operating environment. As the feature and its sub-features are crucial for mobile applications, they are defined as mandatory (and the *coexistence* feature is abstract);

- *Security:* fundamental feature for any educational application. Due to the relevance of the features regarding *integrity* and *confidentiality*, they are defined as mandatory. The *authentication* sub-feature is optional;

- *Communication:* supports the exchange of information among users enabling, for instance, message exchange, delivery of test results and even synchronization of activities performed in other mobile devices. It is an optional feature; and

- *Support:* provides some supporting alternatives to the user such as help and internationalization. It is classified as optional, as well as it sub-features.

B. Validation of the Domain Analysis

Aiming at validating the requirements previously elicited, this phase suggests the application of a checklist to evaluate the main technical issues related to M-SPLearning. An online checklist, composed of twelve multiple-choice questions, was prepared and applied to domain experts. The first two questions were related to the participants' experience in SPL and m-learning. The remaining questions were related to the requirements catalog resultant from the domain analysis.

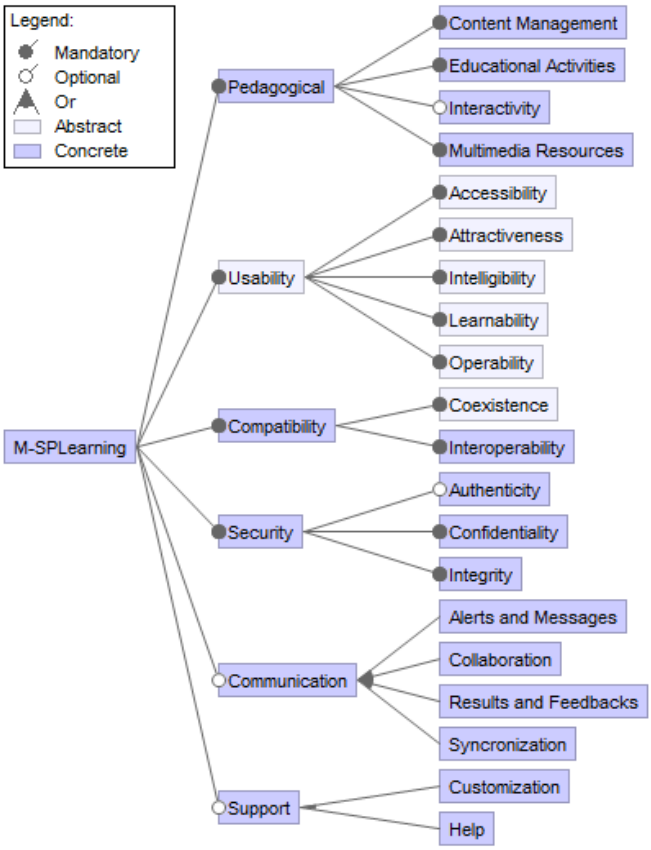

Figure 1. The M-SPLearning Feature Diagram

From the obtained results, taking into account the participants' point of view, the requirements catalog was considered adequate (60.90%) or at least regular (39.10%) for all items evaluated. None of the questions related to the catalog was answered as unsatisfactory.

Despite the positive results achieved, it is important to highlight that the requirements validation conducted is still preliminary. In this scenario, an empirical validation through experiments with qualified practitioners from industry and academia would be extremely relevant. These experiments have been planned and should be performed in short term.

C. Architecture Definition

The third phase involves the definition of a PLA for M-SPLearning. Ultimately, such architecture can serve as a basis for the derivation of all products defined in the scope of the SPL proposed.

In short, we defined a component-based architecture aiming at bringing together the benefits of reuse and modularization to the systematic characteristics of SPLs. To do so, we take into account SMarty [22], a variability management approach for UML-based SPL. More specifically, we applied the UML profile that the approach provides, the SMartProfile, according to a set of guidelines from the SMartyProcess.

According to OliveiraJr et al. [22], the SMartyProfile contains a set of stereotypes and tagged values to represent variability in SPL models. This profile uses a standard object-oriented notation and its profiling mechanism on order to provide an extension of UML and to allow graphical representation of variability concepts. Figure 2 illustrates the M-SPLearning architecture according to SMarty.

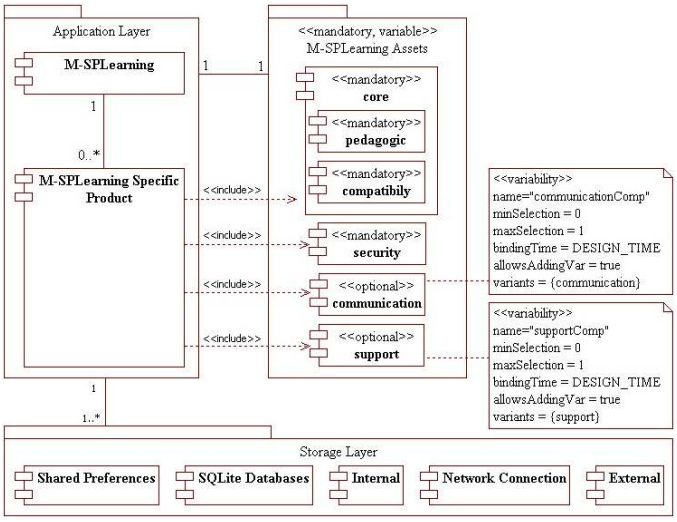

Figure 2. The M-SPLearning Architecture

From the architectural diagram, it is possible to identify the basic structure used in the construction of M-SPLearning. Notice that the assets package contains the components that correspond to the requirements of M-SPLearning. These components provide the concrete features, defined in the feature diagram (Figure 1).

The components that represent the essential features were grouped as the core component. Thus, the idea is to unify all the fundamental modules to any product generated by M-SPLearning. Furthermore, the other components have a dependence relationship with the core, since they need the core to provide their functionalities.

The application layer contains the component that represents M-SPLearning, showing their association with the assets package and making explicit the possibility of creating multiple products. Each generated product uses the components available in the M-SPLearning assets, thereby allowing that different configurations can be used according to the settings defined in application layer.

The products communicate with the storage layer to perform the operations provided by SPL that require data access or data writing. Since the products generated can be manipulated by developers, they can also communicate with the storage layer in an external mode. This alternative can be useful, for instance, if it is necessary to implement a variability that is not supported by M-SPLearning.

Next we summarize the design of the components that implement the concrete features of M-SPLearning. These components define the similarities and variabilities supported by M-SPLearning.

D. Components Design

The design phase completes the process of creating a SPL. According to Krueger [7, 8], variabilities and similarities among specific products of a SPL must be identified. Therefore, the features of the components stored in the repository should be designed and visually represented through a new component diagram, as shown in Figure 3.

From the component diagram, it is possible to explore all the details of the concrete features covered by the SPL, detailing the variation points and identifying each component with its respective SMartyProfile stereotype. These stereotypes make it easier the understanding of the possible configurations of the m-learning applications resulting from M-SPLearning.

The dependence relationship among components becomes explicit as well. As previously said, the core component fully unifies the mandatory features. This is particularly necessary for *security*, *communication* and *support* components.

Notice that the *communication* component is not directly dependent to the *core*. Actually, this dependence is intermediated by the *security* component, guaranteeing that all products have a secure communication. This ensures the architectural obligation that all components depend on the *core* component, which is responsible for providing the fundamental features of M-SPLearning.

M-SPLEARNING: IMPLEMENTATION

In order to preliminary evaluate the application of M-SPLearning, some of its features were implemented according to the SOA architectural pattern, which is frequently associated with the concept of SPL [14, 15, 23].

Basically, M-SPLearning presents a logical view of a SPL defined for the configuration and creation of Android applications. These applications were generated through the consumption of services, and mainly by the collaboration among the implemented modules. Three modules are particularly important, since their interactions and responsibilities characterize our SPL. Figure 4 highlights the adopted architecture, relating it to the initial architecture proposed by M-SPLearning.

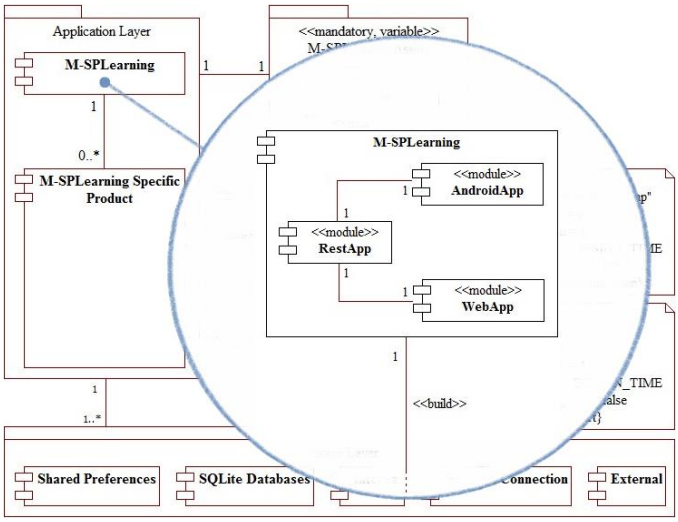

Figure 4. Main modules implemented in the architecture of M-SPLearning

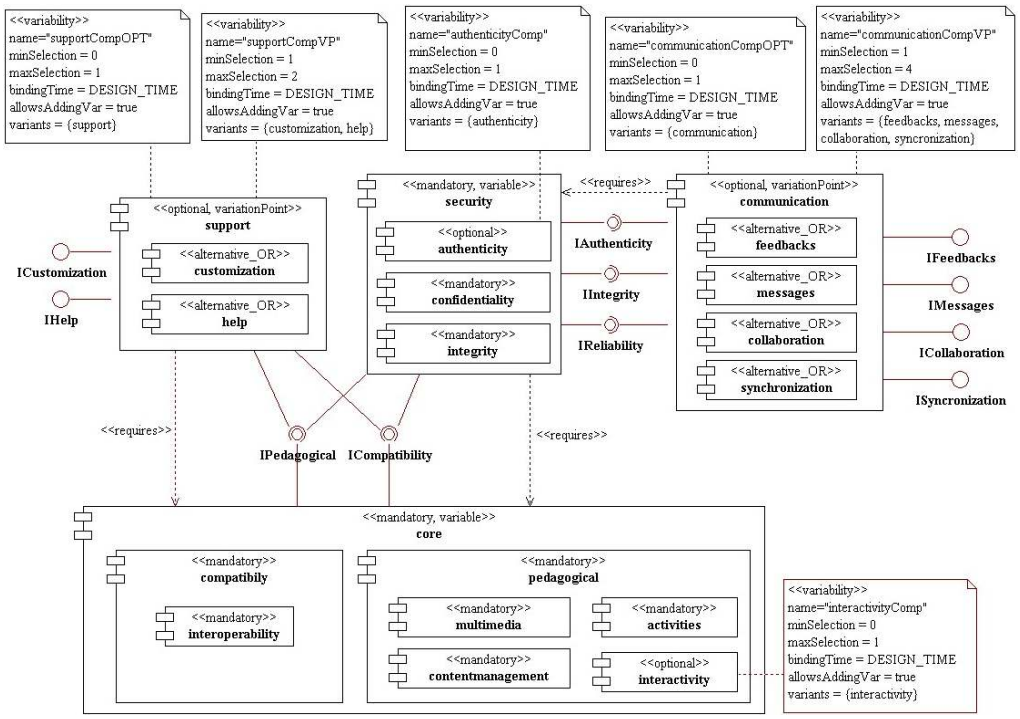

Figure 3. The M-SPLearning Component Diagram

The *RestApp* module is based on the *Representational State Transfer* (REST), an architectural style that is characterized by the use of Web technologies and protocols for the creation and delivery of services [24]. This application accesses a remote database in which all the information of this case study was stored in addition to the features (similarities and variabilities) specified in M-SPLearning and used for the generation of customized products.

A visual interface for creating the products is provided by the *WebApp* application. In this interface, it is possible to configure the variabilities and request the generation of a customized product. This module was implemented only using client-side technologies. The idea was to demonstrate interoperability in a SOA architecture, since *RestApp* module also communicates with an Android module (described next).

Finally, the *AndroidApp* module was developed on the Android platform using the Java programming language. This application allows a service, available in *RestApp* module, executes a custom "build" according to the variabilities configured in *WebApp* application. So, it is possible to generate customized products, which can be installed on Android devices.

The generated product is adapted, in run time, to the variabilities configured at time of its creation. Actually, each application carries its own similarities and variabilities.

In our preliminary implementation, the *compatibility* feature and its sub-feature *interoperability* were built aiming to consumption of REST services. The *security* feature and all its sub-features were also available, allowing secure and reliable authentications. The *interactivity* feature, in turn, allowed the integration with social networks (Facebook and Twitter).

Figure 5 illustrates two products generated by the implementation of M-SPLearning. The first product was generated with only the optional *authentication* feature; the second product was configured in a similar way, but including the *interactivity* feature.

Although not all the features defined by M-SPLearning were implemented yet, it was possible to apply a set of concrete software architectures that altogether characterize a functional SPL, with some of the features presented in this paper.

THREATS TO VALIDITY

Regarding the preliminary validation of M-SPLearning, some threats have been identified:

- *Missing relevant requirements.* The requirements catalog adopted [11] may have omitted important requirements to the m-learning applications domain. This threat was identified since there is no guarantee that the derivation of the original catalog [11] from the ISO [18] included all the features of the target domain.

- *Validation of domain analysis.* In this phase a checklist was proposed. In spite of being answered by experts, this checklist may be not efficient in the case of the proposed questions be not really effective for this validation.

- *Similarities and variabilities.* M-SPLearning is a proposal of SPL, i.e., it has not been fully implemented yet. This can represent a threat, since similarities and variabilities were still not applied in real products, making difficult to evaluate the proposed SPL in practice.

Despite the threats observed, it is important to highlight that the derivation of a PLA from the requirements catalog and the feature diagram, both of them in agreement with the ISO quality standard [18], can significantly increase the quality of M-SPLearning. Controlled and systematic experiments should be conducted in the near future in order to formally validate M-SPLearning and its main aspects.

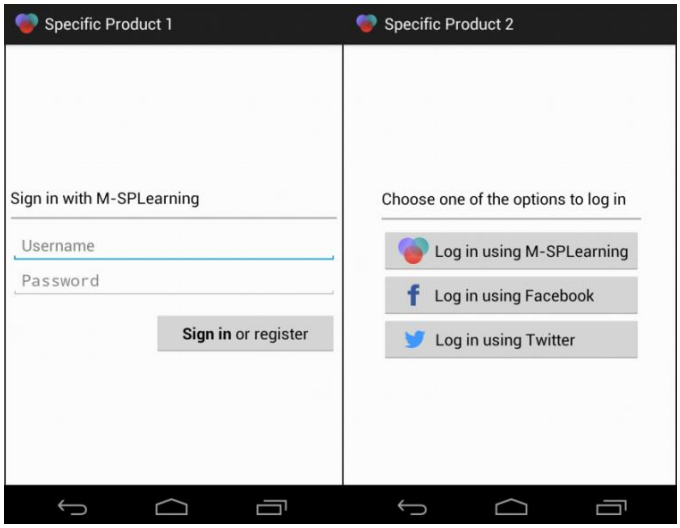

Figure 5. Products generated by the implementation of M-SPLearning

Conclusions and Future Work

In this paper we describe M-SPLearning – a SPL for m-learning applications. M-SPLearning was developed through a process based on concepts and relevant practices of software engineering. Its main contribution relies on providing benefits with regard to the overall quality, domain comprehension, and reduction of the time spent in the development and maintenance of m-learning applications.

As future work, M-SPLearning must be fully implemented and the generated products should be evaluated with regard to quality and compliance. In this sense, we point out the need of conducting a complete evaluation of M-SPLearning. This evaluation has been planned and will require efforts to develop a considerable set of m-learning applications. Quantitative studies involving detailed experiments to measure the effort required to use M-SPLearning should be conducted as well.

Finally, we also intend to investigate other adoption models. For instance, the extractive model can be applied in similar products to those generated by M-SPLearning aiming at increasing the validity of similarities and variabilities specified. The reactive model can be investigated as an alternative to the evolution of the proposed SPL as well.

ACKNOWLEDGMENT

The authors would like to thank the Brazilian funding agencies (FAPESP, CNPq and CAPES) and CEPID-CeMEAI (FAPESP 2013/07375-0) for their financial support. We also thank Cast Informática S.A. for supporting this research.

References

[1] D. Keegan. The incorporation of mobile learning into mainstream education and training. Proccedings of m-Learning 2005 - 4th World Conference on m-learning, 2005.

[2] S. Wexler, J. Brown, D. Metcalf, D. Rogers and E. Wagner. Mobile learning: What it is, why it matters, and how to incorporate it into your learning strategy. Guild Research, 2008.

[3] C. O'Malley, G. Vavoula, J. P. Glew, J. Taylor, M. Sharples and P. Lefrere. Guidelines for learning/teaching/tutoring in a mobile environment. Technical Report, MOBIlearn/UoN/UoB/OU, 2003.

[4] R. C. Durscki, M. M. Spinola, R. C. Burnett and S. S. Reinehr. Software product lines: risks and benefits of your implanting. VI International Symposium on Software Process Improvement, 2004.

[5] F. J. Linden, K. Schmid and E. Rommes. Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[6] P. Clements. Being Proactive Pays Off. IEEE Software - Point / Counter Point, p. 28–31, 2002.

[7] C. Krueger. Easing the Transition to Software Mass Customization. 4th International Workshop on Software Product-Family Engineering, Springer-Verlag, London, UK, p. 282–293, 2002.

[8] C. Krueger. Eliminating the Adoption Barrier. IEEE Software - Point / Counter Point, p. 28–31, 2002.

[9] F. Ozdamli and N. Cavus. Basic elements and characteristics of mobile learning. World Conference on Educational Technology Researches, Volume 28, p. 937–94, 2011.

[10] A. Schepman, P. Rodway, C. Beattie and J. Lambert. An observational study of undergraduate students' adoption of (mobile) note-taking software. Computers in Human Behavior, 28(2), 308-317, 2012.

[11] N. F. Duarte Filho and E. F. Barbosa. A Requirements Catalog for Mobile Learning Environments. SAC'13 Proceedings of the 28th Annual ACM Symposium on Applied Computing, 2013.

[12] N. Taylor, N. Medvidovic and E. M. Dashofy. Software Architecture: Foundations, Theory, and Practice. John Wiley & Sons, USA, 2009.

[13] D. L. Dalmon, L. O. Brandão, A. F. Anarosa and S. Isotani. A Domain Engineering for Interactive Learning Modules. Journal of Research and Practice in Information Technology. v. 44, p. 309–330, 2012.

[14] F. G. Marinho, R. M. C. Andrade, C. Werner, W. Viana, M. E. F. Maia, L. S. Rocha, E. Teixeira, J. B. F. Filho, V. L. L. Dantas, F. Lima and S. Aguiar. MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications, Science of Computer Programming, v. 78, p. 2381–2398, 2013.

[15] F. G. Marinho, A. L. Costa, F. F. P. Lima, J. B. B. Neto, J. B. F. Filho, L. Rocha, V. L. L. Dantas, R. M. C. Andrade, E. Teixeira and C. Werner. An architecture proposal for nested software product lines in the domain of mobile and context-aware applications. Proceedings - 4th Brazilian Symposium on Software Components, Architectures and Reuse, SBCARS 2010, p. 51–60, 2010.

[16] G. G. Pascual, M. Pinto and L. Fuentes. Component and aspect-based service product line for pervasive systems. CBSE'12 - Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering, p. 115–124, 2012.

[17] R. Llamas, R. Reith and M. Shirer. Android and iOSCombine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year. IDC - Press Release, 2013. Available in: http://goo.gl/WsBRBh.

[18] ISO/IEC JTC 1/SC 7. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011. Available in: http://goo.gl/tkpwPa.

[19] J. van Gurp, J. Bosch and M. Svahnberg. On the notion of Variability in Software Product Lines. Working IEEE/IFIP Conference on Software Architecture (WICSA 2001), 2001.

[20] J. Bosch. Software Product Lines: Organizational Alternatives. International Conference on Software Engineering (ICSE'01), 2001.

[21] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Carnegie-Mellon University Software Engineering Institute, Pittsburgh, Pennsylvania, 1990.

[22] E. A. OliveiraJr, I. M. S.Gimenes and J. C. Maldonado. Systematic Management of Variability in UML-based Software Product Lines. In: J. UCS, vol. 16, num. 17, pp. 2374–2393, 2010.

[23] A. S. Nascimento, C. M. F. Rubira and J. Lee. An SPL approach for adaptive fault tolerance in SOA. In Proceedings of the 15th International Software Product Line Conference (SPLC '11), vol.2, art. 15 , pp. 1–8, New York, 2011.

[24] R. T. Fielding. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Dissertation. University of California, Irvine, 2000.