Journal of Universal Computer Science, vol. 28, no. 10 (2022), 1058-1086
submitted: 4/2/2020, accepted: 8/10/2022, appeared: 28/10/2022 CC BY-ND 4.0

Development and Evaluation of a Software Product Line for M-Learning Applications

Venilton FalvoJr
(University of São Paulo, São Carlos, São Paulo, Brazil
https://orcid.org/0000-0003-2367-3761, falvojr@usp.br)

Anderson da Silva Marcolino
(Federal University of Paraná, Palotina, Paraná, Brazil
https://orcid.org/0000-0002-4014-1882, anderson.marcolino@ufpr.br)

Nemesio Freitas Duarte Filho
(Federal Institute of São Paulo, Sertãozinho, São Paulo, Brazil
https://orcid.org/0000-0001-5084-3733, nemesio@ifsp.edu.br)

Edson OliveiraJr
(State University of Maringá, Maringá, Paraná, Brazil
https://orcid.org/0000-0002-4760-1626, edson@din.uem.br)

Ellen Francine Barbosa
(University of São Paulo, São Carlos, São Paulo, Brazil
https://orcid.org/0000-0003-3275-2293, francine@icmc.usp.br)

Abstract: The popularity of mobile devices in all social classes has motivated the development of mobile learning (m-learning) applications. The existing applications, even having many benefits and facilities in relation to the teaching-learning process, still presents problems and challenges, especially regarding the development, reuse and architectural standardization. Particularly, there is a growing adoption of the Software Product Line (SPL) concept, in view of research that investigates these gaps. This paradigm enables organizations to explore the similarities and variabilities of their products, increasing the reuse of artifacts and, consequently, reducing costs and development time. In this context, we discuss how systematic reuse can improve the development of solutions in the m-learning domain. Therefore, this work presents the design, development and experimental evaluation of M-SPLearning, an SPL created to enable the systematic production of m-learning applications. Specifically, the conception of M-SPLearning covers from the initial study for an effective domain analysis to the implementation and evaluation of its functional version. In this regard, the products have been experimentally evaluated by industry software developers, providing statistical evidence that the use of our SPL can speed up the time-to-market of m-learning applications, in addition to reducing their respective number of faults.

Keywords: software product lines, mobile learning applications, variability management, experimental evaluation
Categories: D.2.1, D.2.5, D.2.8, D.2.10, D.2.11, D.2.13, L.2.2, L.3.0, L.7.0
DOI: 10.3897/jucs.90663

1 Introduction

In the last decades, several changes in software reuse approaches have led to the concept of Software Product Lines (SPL). According to [Linden et al., 2007], this approach represents one of the most disruptive paradigms in software development since the advent of high-level programming languages. Thereby, companies which have developed project-by-project software for years, can now focus on building and maintaining this systematic reuse strategy, focusing on the variabilities of their products.

For successful adoption of an SPL, its domain must be carefully defined. If the domain is too large and the products vary widely, the core assets will be overloaded beyond their designed variabilities. As a result, production savings will be lost and the SPL will collapse. Therefore, properly defining the domain and its respective variabilities is essential. For this, the concept of variability management [Chen and Babar, 2011, Galster et al., 2014] arises intrinsically, aiming to enable the healthy diversification of a domain's product portfolio [Capil la et al., 2013].

On the other hand, the rapid growth of Information and Communication Technologies (ICT) has favored the emergence of innovative ways of dealing with the problems of traditional education [Kraut, 2013]. More than ever, people, companies and organizations are looking for effective ways of training and qualification, in view of an increasingly competitive ICT job market [Kukulska-Hulme and Traxler, 2005, Sharples et al., 2009, Matzavela and Alepis, 2021]. Mobile learning (m-learning), for instance, has provided a strong interaction between learners and instructors, enabling them to actively participate in the knowledge construction process anytime and anywhere [Moreira and Rocha, 2018, Matzavela and Alepis, 2021].

Despite the benefits provided in the teaching and learning process, the m-learning solutions still presents challenges in its adoption globally [Sharples, 2013, Kraut, 2013]. Few studies discuss public policies and definitions of a base curriculum to guide the effective adoption of m-learning applications. Furthermore, propositions of reuse strategies in this domain are scarce, making it a relevant research object [FalvoJr, 2015, Marcolino and Barbosa, 2015, Marcolino and Barbosa, 2016].

Therefore, there is a lack of studies that present m-learning solutions developed through systematic reuse strategies, such as SPL. [Bezerra et al., 2009] and [Chen and Babar, 2011] conducted broader systematic reviews and both identified gaps related to SPL adoption and variability management, respectively. Furthermore, although the concept of SPL is already consolidated in some domains, there are few studies exploring m-learging applications specifically [FalvoJr, 2015, Marcolino and Barbosa, 2015, Marcolino and Barbosa, 2016].

Such gaps, identified through formal literature reviews in the domain of m-learning applications, motivated us to develop [FalvoJr et al., 2014a, FalvoJr et al., 2014b] and evaluate (focus of this work) an SPL in this domain. The SPL is named `M-SPLearning` and it has been created based on a concise UML-based variability management approach, named SMarty (**S**tereotype-based **M**anagement of V**ar**iabili**ty**), which provides mechanisms to facilitate the identification and representation of variabilities [OliveiraJr et al., 2010, Marcolino et al., 2013, Marcolino et al., 2014b, Marcolino et al., 2014a, Bera et al., 2015, Marcolino et al., 2017].

It is worth mentioning that `M-SPLearning` differs significantly from other studies focused on Generative Learning Objects (GLO) [Costea et al., 2018, Burbaite et al., 2014, Stuikys et al., 2013]. While a Learning Object (LO) concept usually refers to a small-sized, reusable instructional component, designed for distribution over the Internet, for use in different Learning Management Systems (LMS), being accessible by many users [Costea et al., 2021, Anido-Rifon et al., 2001] and a GLO are reusable educational templates to be filled with learning content [Costea et al., 2021], `M-SPLearning` provides a mobile platform that can support learning content, created through the platform itself by their users. To conclude, considering the technical aspects of an SPL, it could be evolved to support and integrate GLO and LO, but in this study, only the aspects as a LMS are considered.

In this paper we discuss how the adoption of an SPL approach can improve the development of m-learning applications. For this, we experimentally evaluate `M-SPLearning`, comparing it with a single software development methodology, common in the industry. The research question to be answered is: ***"Can SPLs improve the time-to-market of stable m-learning applications?"***

The experimental evaluation is based on two relevant software development variables: time-to-market and number of faults. These variables can be directly influenced by the adopted development methodology and approaches that support the variabilities and commonalities management [Hubaux et al., 2011, Capil la et al., 2013]. Additionally, in the industry segment, on-time delivery and quality are crucial variables for business success and customer satisfaction, both are important facets of product development strategy [Hubaux et al., 2011, Dóra et al., 2013].

In this perspective, the experiment was conducted in person at a Brazilian software company, aiming at comparing the software development methodology used by its employees with `M-SPLearning`. In addition to comparing both approaches, it was also possible to promote the exchange of experiences between experts from industry and academia, in view of the low adoption of SPL in the Brazilian industry in general [FalvoJr, 2015, Marcolino and Barbosa, 2015, Marcolino and Barbosa, 2016].

The paper is organized as follows: Section 2 presents the `M-SPLearning` and encompasses essential background; Section 3 addresses the `M-SPLearning` experimental evaluation; Section 4 discusses the lessons learned from the development and application of `M-SPLearning` and its prospective improvements; Section 5 summarizes the related work; and Section 6 presents our conclusions and perspectives for future work.

2 M-SPLearning: an Overview

Mobile learning stands out for its ability to promote genuine interaction between students and instructors, anytime and anywhere, providing exciting new teaching opportunities [Kukulska-Hulme and Traxler, 2005, Sharples et al., 2009]. Despite its benefits, m-learning is still considered an incipient concept, having limitations that hamper its effective development and adoption. For instance, even with the increasing demand for m-learning applications, few studies have addressed development issues through a systematic reuse strategy, such as SPL, in the m-learning domain [FalvoJr, 2015, Marcolino and Barbosa, 2015, Marcolino and Barbosa, 2016].

More specifically, mobile applications can be implemented on different development platforms, mostly Android or iOS. Actually, the domain of m-learning also includes such Operating Systems (OS) and their particularities. Thus, to define a viable domain in terms of scope, we selected a single OS, the Android. Our choice was mainly based on the number of devices that each OS controls, as Android has (consistently) held between 70% and 80% of the global smartphone market share in recent years [Chau and Reith, 2021, StatCounter, 2021].

Considering the ubiquity of mobile devices and the lack of systematic reuse approaches that relate the concepts of mobility and education [FalvoJr, 2015, Marcolino and

Barbosa, 2015, Marcolino and Barbosa, 2016], we have worked on the development of an SPL for the m-learning domain, named `M-SPLearning` [FalvoJr et al., 2014a, FalvoJr et al., 2014b]. Its conception followed the proactive approach proposed by [Krueger, 2002], which includes the phases of Domain Engineering, Architecture and Design. These phases are summarized below, along with the `M-SPLearning`'s product generation dynamics.

2.1 Domain Engineering

The proactive approach is appropriate when the requirements for a set of products to be created are stable and can be defined in advance [Krueger, 2002]. In this context, our SPL was based on the m-learning requirements catalog proposed by [Duarte Filho and Barbosa, 2013] and the quality model ISO/IEC 25010 [ISO/IEC, 2010]. This catalog was established from the results of a systematic review also conducted in the domain of this study.

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, 2013].

We used this requirements catalog as a reference for the creation of an adherent feature model for `M-SPLearning` (Figure 1), designed using FeatureIDE[1]. The interpretation of the feature model is simple and follows the traditional concepts conceived by the Feature-Oriented Domain Analysis (FODA) [Kang et al., 1990]. Basically, each requirement was mapped as a primary feature, generating its respective secondary features. Thus, a total of 30 features were modeled and submitted to the S.P.L.O.T.[2], which indicated 3,840 possible variations for our feature model. Our primary features and their responsibilities for m-learning applications are detailed following:

- *Compatibility*: includes the *Coexistence* and *Interoperability*. Both are mandatory for m-learning applications, given the need for multiple devices to interact in the same ecosystem, which often interoperates with other systems and solutions. In particular, *Coexistence* is intrinsic in the mobile domain, so it was defined as abstract;
- *Pedagogical*: it has the educational and pedagogical requirements, providing the main features of m-learning applications. In particular, the *Interactivity* is optional and depends on the constraint: *(Collaboration **or** ResultsAndFeedbacks **or** Help) **implies** Interactivity*. Lastly, *Content Management*, *Educational Activities* and *Multimedia Resources* are mandatory; the latter offers a choice of one or more multimedia resources to support teaching;
- *Security*: this is a critical feature because any mobile app must send/receive information securely. Subfeatures *Integrity* and *Confidentiality* were defined as mandatory, as they are essential for data consistency and integrity. On the other hand, the *Authenticity* is optional because not every m-learning application has explicit authentication of its users;

[1] Tool available at https://featureide.github.io.
[2] Tool available at http://splot-research.org.

– *Usability*: addresses the essential visual interface features of m-learning applications, abstracting the User Interface (UI) and User Experience (UX) platform standards. It is fundamental for the acceptance of the app in the market, because the products generated by SPL must adopt usability standards that offers an attractive experience to the end users. Therefore, all features were defined as mandatory, with the exception of *Attractiveness*, which has a constraint: *(Audio* ***or*** *Image* ***or*** *Video)* ***implies*** *Attractiveness*. Furthermore, all subfeatures are abstract because the guidelines are provided by the development platform itself, such as Material Design for Android;

– *Communication*: responsible for exchanging information between users, making the teaching ecosystem more collaborative and cohesive, due to the interactivity of learners and synchronization of activities. Anyway, this feature and subfeatures are optional and accept any possible combinations;

– *Support*: offers interesting features for some m-learning applications, such as user support and internationalization. In this context, all elements were classified as optional because they are not mandatory for all applications.

2.2 Architecture

Based on Domain Engineering phase, we designed a software architecture adherent to the requirements m-learning domain. Such architecture and its components represent, in an abstract level, the core assets of `M-SPLearning`. From this moment, SPL variability management has become essential for the organization and standardization of `M-SPLear ning`.

Therefore, the SMarty approach was assigned for this responsibility, mainly because: (i) full compliance with UML, which standardizes SPL design and validation; (ii) cognitive ease in view of compatibility with market modeling tools; and (iii) the existence of experimental evidence of its superior effectiveness compared to other UML-based approaches [Marcolino et al., 2013, Marcolino et al., 2014b, Marcolino et al., 2014a, Bera et al., 2015, Marcolino et al., 2017].

In this context, one of the `M-SPLearning` most representative assets is the architecture diagram (Figure 2). This diagram abstracts the variabilities, similarities and interactions between `M-SPLearning`'s architectural components. So, we have an overview of the SPL and its domain assets, essential information for the Design phase that details the responsibility of each component.

Based on the architectural diagram, it is possible to identify the structural, base used to the construction of the `M-SPLearning`. The package *Core Assets* comprises the SPL concrete features and its components represent the specific features of the m-learning domain, grouped into component *core*. Thereby, the fundamental modules for products generated by `M-SPLearning` could be visually unified. Note that SMarty is used to represent the variabilities present in two components. To conclude, the *Application Layer* contains a component that characterizes the `M-SPLearning` UI, identifying its association with *Core Assets*, enabling the derivation of products. Each product should include the components available in the *Core Assets*, making each m-learning application able to use different features according to its configurations.

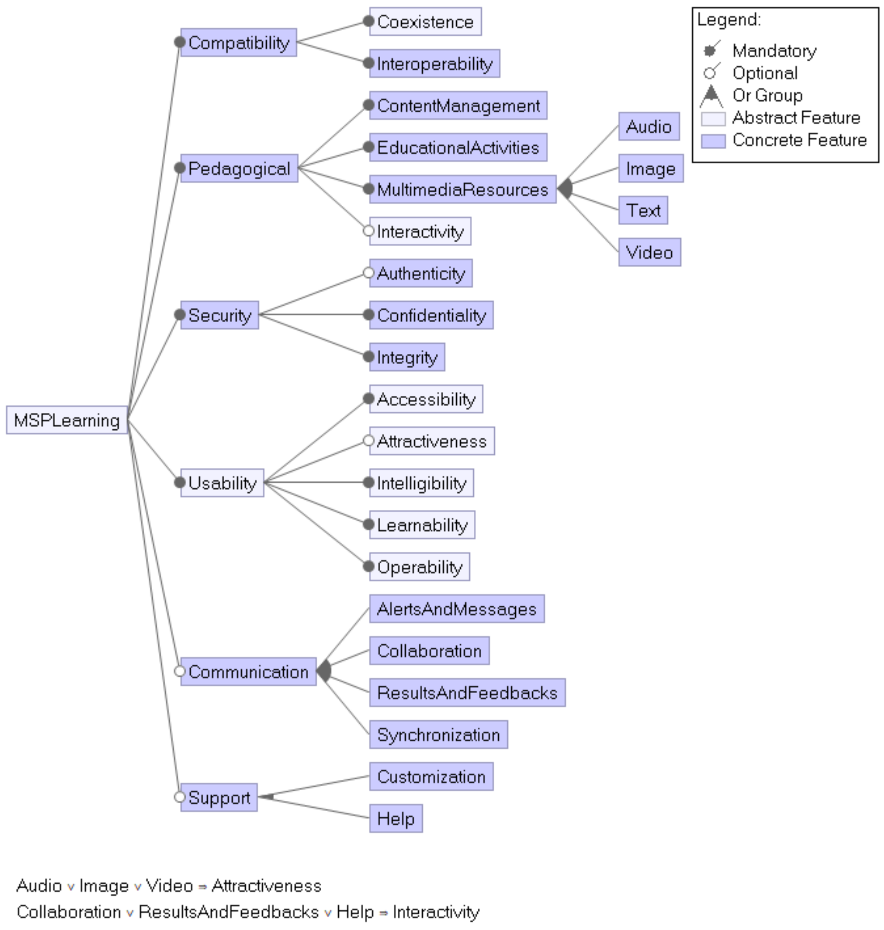


Figure 1: `M-SPLearning` Feature Model (designed using FeatureIDE)

2.3 Design

In this phase of project, the variabilities and similarities identified in the previous phases are designed. For this, the core asset elements were visually represented by another SMarty-based component diagram (Figure 3). This diagram presents all the concrete features modeled for `M-SPLearning`, so it is less abstract than our architectural diagram. The resulting components were tagged with SMarty stereotypes, which allows the analysis of the number of possible configurations of products supported by SPL. In this sense, SMarty provides the following stereotypes: *optional*, *mandatory*, *variable*, *variationPoint* and *variability*.

From this component diagram, we proposed a production plan that introduce the interactions between *UI*, *API Services* and *Android Template*, which together define the configuration/generation dynamics of `M-SPLearning` products. Therefore, we designed a SMarty-based activity diagram to formally describe our Production Plan (Figure 4).

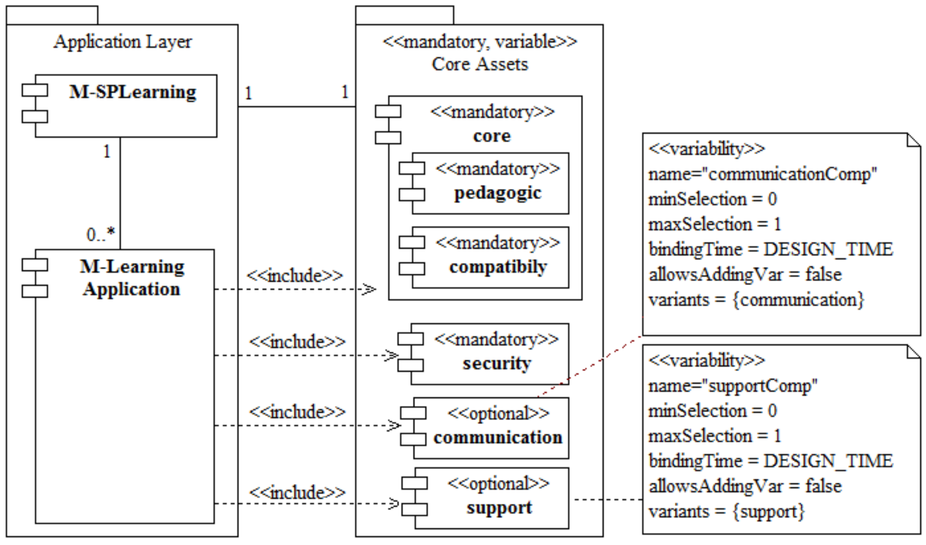


Figure 2: `M-SPLearning` Architecture.

In that regard, the variation point exposes all variabilities elicited for `M-SPLearning`. Thus, the customization of the products was centralized at a single point, simplifying the configuration and generation processes of m-learning applications.

2.4 Product Generation

The creation of products essentially depends on the implementation of the assets/components elicited in the previous phases. Because of this, a specific set of features was prioritized, aiming to define a viable scope for the construction and evaluation of the `M-SPLearning`.

In this context, the following features were implemented, due their importance in the domain of m-learning applications: (i) *Pedagogical*: includes educational activities through the management of interactive and multimedia content; (ii) *Security*: provides confidentiality and integrity of data, which includes the user authentication; and (iii) *Communication*: related to data synchronization. Such features are rendered in the `M-SPLearning`'s UI, which enables the selection of variabilities, through a Website, for product generation (Figure 5).

The products generated are Android Apps ready for installation (APK files), which solve their variability in configuration time. Figure 6 shows same product configurations, highlighting *Pedagogical* variabilities:

(a) ***Interactivity* variability unchecked**, resulting in an authentication provided only by the `M-SPLearning` itself;

(b) ***Interactivity* variability checked**, resulting in the possibility of authentication via social media (Facebook and Twitter);

(c) **Only *Video* feature checked on *Multimedia Resources***, resulting in the exclusivity of this type of content.

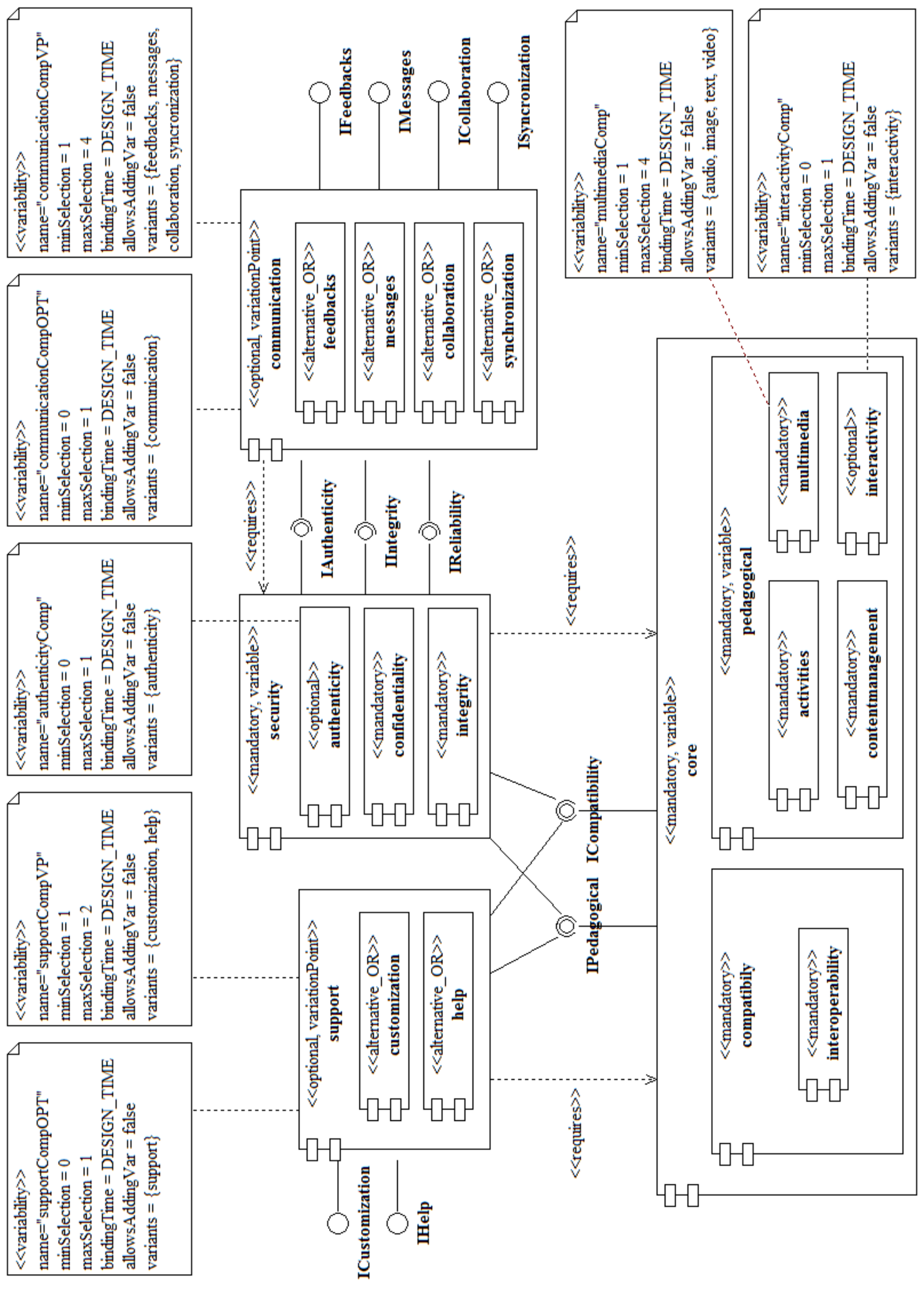

<<variability>>
name="communicationCompVP"
minSelection = 1
maxSelection = 4
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = {feedbacks, messages, collaboration, syncronization}
<<variability>>
name="communicationCompOPT"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = {communication}
<<variability>>
name="authenticityComp"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = {authenticity}
<<variability>>
name="supportCompVP"
minSelection = 1
maxSelection = 2
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = {customization, help}
<<variability>>
name="supportCompOPT"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = {support}
<<optional, variationPoint>> communication
<<alternative_OR>> feedbacks
<<alternative_OR>> messages
<<alternative_OR>> collaboration
<<alternative_OR>> synchronization
IFeedbacks
IMessages
ICollaboration
ISyncronization
<<requires>>
IAuthenticity
IIntegrity
IReliability
<<mandatory, variable>> security
<<optional>> authenticity
<<mandatory>> confidentiality
<<mandatory>> integrity
<<requires>>
ICompatibility
IPedagogical
<<optional, variationPoint>> support
<<alternative_OR>> customization
<<alternative_OR>> help
<<requires>>
ICustomization
IHelp
<<variability>>
name="multimediaComp"
minSelection = 1
maxSelection = 4
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = {audio, image, text, video}
<<variability>>
name="interactivityComp"
minSelection = 0
maxSelection = 1
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = {interactivity}
<<mandatory, variable>> core
<<mandatory, variable>> pedagogical
<<mandatory>> multimedia
<<optional>> interactivity
<<mandatory>> activities
<<mandatory>> contentmanagement
<<mandatory>> compatibiliy
<<mandatory>> interoperability

Figure 3: M-SPLearning Design.

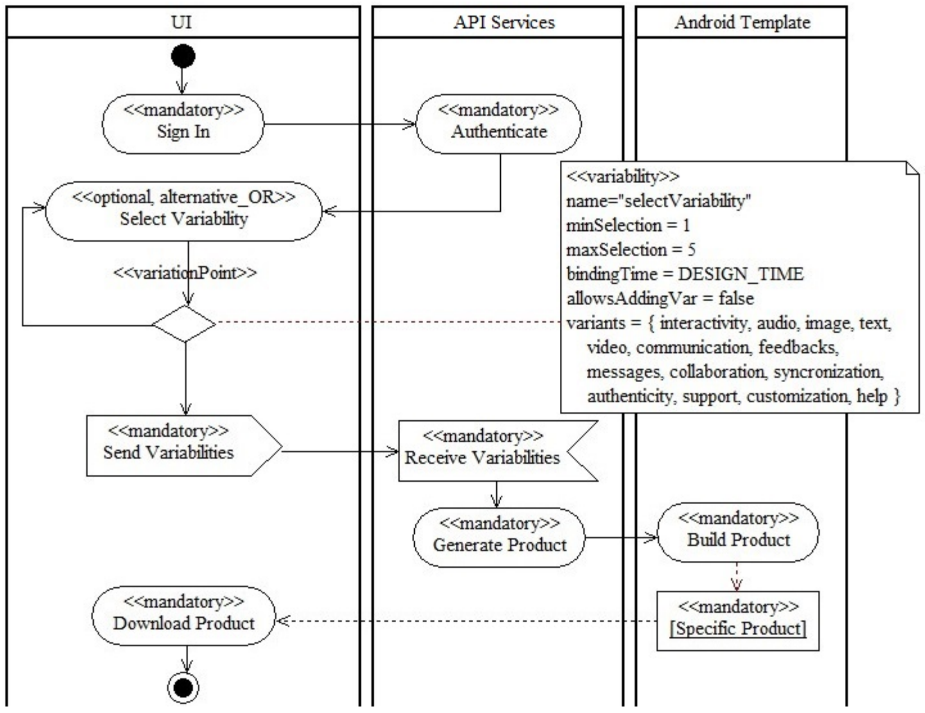

UI
API Services
Android Template
<<mandatory>> Sign In
<<mandatory>> Authenticate
<<optional, alternative_OR>> Select Variability
<<variationPoint>>
<<variability>>
name="selectVariability"
minSelection = 1
maxSelection = 5
bindingTime = DESIGN_TIME
allowsAddingVar = false
variants = { interactivity, audio, image, text, video, communication, feedbacks, messages, collaboration, syncronization, authenticity, support, customization, help }
<<mandatory>> Send Variabilities
<<mandatory>> Receive Variabilities
<<mandatory>> Generate Product
<<mandatory>> Build Product
<<mandatory>> [Specific Product]
<<mandatory>> Download Product

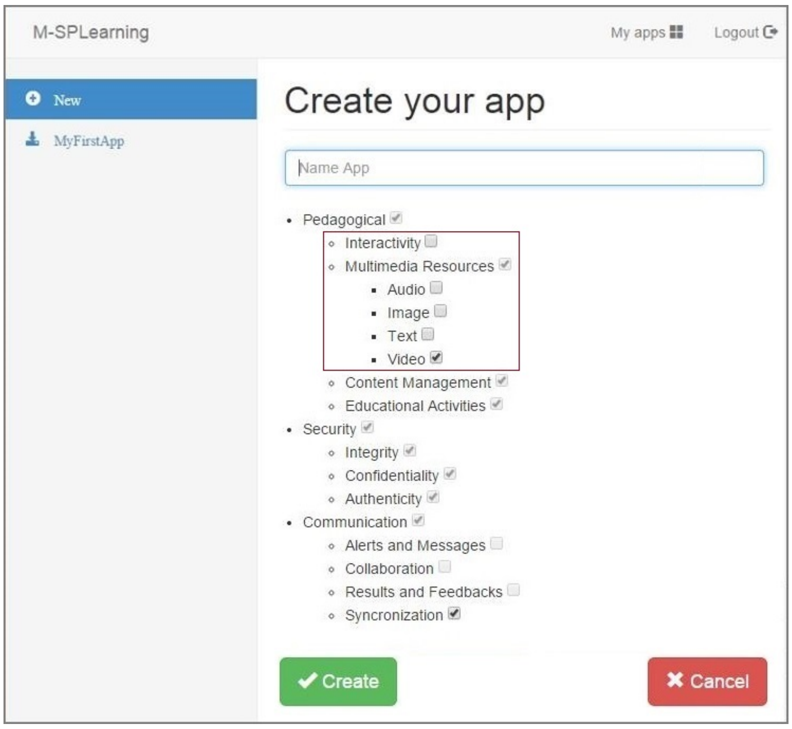
Figure 4: `M-SPLearning` Production Plan.

To conclude, the conception of `M-SPLearning` involved analysis, design and construction of an SPL for generating m-learning applications. In particular, an experimental evaluation was conducted, which represents the main contribution of this work. Details about the experiment performed and the results obtained are presented in Section 3.

3 M-SPLearning Experimental Evaluation

This section reports the `M-SPLearning`'s experimental evaluation, which observes the time-to-market and the quality of the generated products (m-learning applications). Our SPL was compared with an ad hoc methodology, which we named as Single Software Development (SSD). This methodology occurs when developers use only their own knowledge to implement m-learning applications from scratch, without any reuse technique.

This experimental evaluation considered the analysis and comparison of two important indicators involved in the software development process: time-to-market and quality (number of faults). Although the comparison does not consider two SPL, the final objective is to verify if the core assets components are enough to allow the development of stable m-learning apps. Additionally, we would like to obtain insights favorable to the adoption of SPL in the m-learning domain, promoting the demystification of this systematic reuse approach for the creation of educational applications, in the context of software development companies, aiming to bring academia and industry closer together.



M-SPLearning | My apps | Logout

- New
- MyFirstApp

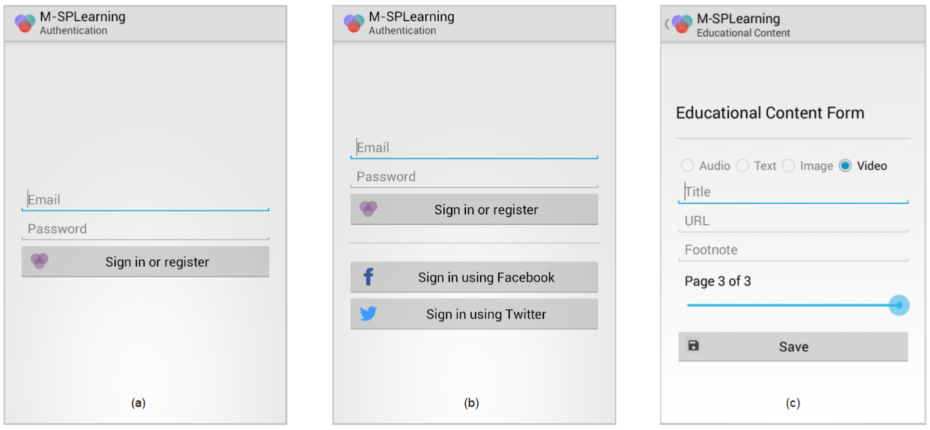
Create your app

Name App

- Pedagogical ✓
  - Interactivity ☐
  - Multimedia Resources ✓
    - Audio ☐
    - Image ☐
    - Text ☐
    - Video ✓
  - Content Management ✓
  - Educational Activities ✓
- Security ✓
  - Integrity ✓
  - Confidentiality ✓
  - Authenticity ✓
- Communication ✓
  - Alerts and Messages ☐
  - Collaboration ☐
  - Results and Feedbacks ☐
  - Syncronization ✓

✔ Create | ✖ Cancel

Figure 5: `M-SPLearning` UI: highlighting Pedagogical variabilities.



(a) M-SPLearning — Authentication

Email

Password

Sign in or register

(b) M-SPLearning — Authentication

Email

Password

Sign in or register

Sign in using Facebook

Sign in using Twitter

(c) M-SPLearning — Educational Content

Educational Content Form

Audio ○ Text ○ Image [● Video]

Title

URL

Footnote

Page 3 of 3

Save

Figure 6: `M-SPLearning` Products - Pedagogical Variations: (a) Interactivity unchecked; (b) Interactivity checked; and (c) Only Video checked on Multimedia Resources.

For this, the experimental process was carried out in the Brazilian software industry, taking into account the knowledge of employees/participants. Thus, the results of each methodology allowed us to verify in a second instance, as lessons learned, how the developers behaved when exploring a totally new approach and outside their comfort zone. Ultimately, we assume that the SSD approach could represent a bias, due to the participants' experience with this methodology in their day-to-day work. Therefore, the experimental evaluation will also contribute to the discussion of such nuances.

We followed the guidelines proposed by [Wohlin et al., 2012] and the report template suggested by [Jedlitschka and Pfahl, 2005] for the conduction of this experiment.

3.1 Aim and Research Question

The experiment aimed at **comparing** `M-SPLearning` (also referenced as SPL in this section for convenience) with an SSD, **for the purpose of** identifying the most efficient, **with respect to** the time spent on the creation of m-learning applications and the number of faults found, **from the point of view** of software engineers **in the context of** practitioners from a software development company.

Two research questions (R.Q.) based on the research objective were raised:

– **R.Q.1** Which methodology is more efficient regarding time-to-market in the m-learning domain: SSD or SPL?

– **R.Q.2** Which methodology presents more quality, in terms of number of faults, in the m-learning domain: SSD or SPL?

3.2 Experimental Planning

This section describes the experimental planning and procedures for supporting future replications. All information regarding the design, configuration of the environment, dynamics of execution and tests are in the experimental package, available at http://falvojr-msc.github.io/msplearning.

Formulation of Hypotheses

We defined two sets of hypotheses to be tested. Each of them is related to its respective research questions (R.Q.1 and R.Q.2):

R.Q.1 hypotheses: time-to-market

– **Null Hypothesis (H_0)**: there is no significant difference of time-to-market (t) between SSD and `M-SPLearning`. ($H_0 : \mu(t(SSD)) = \mu(t(SPL))$);

– **Alternative Hypothesis (H_1)**: there is a significant difference of time-to-market (t) between SSD and `M-SPLearning`. ($H_1 : \mu(t(SSD)) \neq \mu(t(SPL))$).

R.Q.2 hypotheses: quality, in terms of number of faults

– **Null Hypothesis (H_0)**: there is no significant difference between SSD and `M-SPLearning` with regard to quality, in terms of number of faults (f), in the software products created. ($H_0 : \mu(f(SSD)) = \mu(f(SPL))$);

– **Alternative Hypothesis (H_1)**: there is a significant difference between SSD and `M-SPLearning` with regard to quality, in terms of number of faults (f), in the software products created. ($H_1 : \mu(f(SSD)) \neq \mu(f(SPL))$).

Objects

We considered two software products configurations for m-learning applications using Android platform for evaluating SSD and `M-SPLearning`: Product 1 (P1) and Product 2 (P2). They are described as follows:

– P1: m-learning application that provides support to image-based educational content;

– P2: m-learning application that provides support to video-based educational content.

Selection of Variables

The dependent variables time (t) and faults (f) were defined by the following equations (Equation 1 and Equation 2):

$$\mu(t) = (\Sigma xi)/n, i = 1..n \qquad (1)$$

$$\mu(f) = (\Sigma yi)/n, i = 1..n \qquad (2)$$

where:

t is the implementation time (minutes);

f is the number of faults;

xi is the time of implementation of participant i;

yi is the number of faults detected in the implementation of participant i;

n is the total of participants in the experiment.

Independent variables are the development methodology, which is a factor with one control (SSD) and one treatment (SPL), and the software product configuration for m-learning application, which is a factor with two treatments (P1 and P2). Table 1 shows the description of dependent and independent variables.

Time-to-market is the time spent, in average, for the implementation of a software product with a specific group of variabilities of `M-SPLearning`. Regarding the number of faults, the implemented products were tested using the concept of test cases [Craig and Jaskiel, 2002]. In this way, we quantified the average number of defects for each application/product. Such metrics are relevant since they are directly related to time-to-market and quality of the m-learning applications. Additionally, they are relevant metrics for development industry [Jabangwe et al., 2018].

Selection of Participants

The experiment was carried out with employees of a Brazilian software development company, who had 2.72 (average) years of experience as developers in the Java programming language (target platform of the experiment on the Android platform). In this context, 18 developers participated as volunteers within the company environment. We emphasize that the participants also have experience in architecture and software design.

Variable (Type)	**Class**	**Entity**	**Type of Attribute**	**Scale Type**	**Range**	**Counting Rule**
Development Methodology (Independent)	Method	Software development methodology	N/A	Nominal	SSD and SPL	N/A
Software Products (Independent)	Product	Mobile software product	N/A	Nominal	P1 and P2	N/A
Time (Dependent)	Product	Time to market	Internal: time; external: time to market.	Ordinal	From 00:00:00 to 03:00:00	Eqn. (1)
Faults (Dependent)	Product	Number of faults	Internal: faults; external: quality.	Ordinal	Any positive integer	Eqn. (2)

Table 1: Description of Dependent and Independent Variables.

The number of practitioners available in the industry limits a random selection of participants from a population. To reduce this bias, the random capacity was applied at instrumentation level, more specifically, at the assignment of the development methodology (SSD or SPL) and software product (P1 or P2) by participant.

Block classification was defined by two factors with two treatments, which were interspersed in four groups. The block classification avoid undesirable effects in the treatments comparison, rising the experimental precision and the balancing, besides to simplify and improve the statistical analysis of the experimental data [Brooks et al., 1996]. The balancing was applied in the tasks, which were assigned in equal numbers to a similar number of participants. Therefore, the 18 participants were randomly separated into groups:

- **First Group:** focused on SSD with P1 and `M-SPLearning` with P2;
- **Second Group:** focused on `M-SPLearning` with P1 and SSD with P2;
- **Third Group:** focused on SSD with P2 and `M-SPLearning` with P1; and
- **Fourth Group:** focused on `M-SPLearning` with P2 and SSD with P1;

Instrumentation

The experiment was supported by the following set of instruments: (i) similar desktop computers with all necessary tools (Eclipse IDE and plugins); (ii) a consent term for the experimental study; (iii) a characterization questionnaire; (iv) use case, component and sequence UML diagrams; (v) interface messages; (vi) database model; (vii) a project base; (viii) similarities of the products; and (ix) experimental forms for SSD and `M-SPLearning`, randomly distributed and feedback questionnaire.

Analysis Procedures

The main assessment tools were the products developed based on two software specifications (P1 and P2) for m-learning applications, also available in the experimental package.

A specific niche of features was used for the evaluation. The variabilities related to multimedia resources enabled the creation of up to 15 different products. P1 and P2 were specified and implemented by SSD and SPL methodologies.

To collect the data for the analysis of time-to-market, the initial and final time of implementation process for P1 and P2 were registered individually in the experimental form to be calculated in Equation 1. On the other hand, for the analysis of quality, each of 15 developed products were tested and the number of faults was collect to compare the use of SPL and SSD methodologies by means of the Equation 2.

3.3 Execution

This section describes how the experiment was conducted. Firstly, it is worth mentioning that the execution in the industry was an important project decision, aiming to bring Brazilian software development companies closer to reuse approaches already consolidated in the academy, such as SPL.

Sampling

The sample was composed of a total of 21 practitioners, who participated in the training session. However, 18 participants, effectively contributed in the experimental execution, due to the unavailability of three volunteers in the execution day.

Pilot Project

A pilot project was developed with two practitioners from industry, who evaluated the study instrumentation and established the duration of the training and execution sessions. The results and these participants were not considered in the final execution and data analysis of the experiment.

Training

The participants underwent a three-day training session, in which were considered the essential concepts of Android development for SSD and `M-SPLearning` with the Eclipse IDE. The knowledge was evaluated through essays at the end of each training session. On the fourth day, the experiment was performed. The training time lasted approximately 40 minutes. The materials used were: (1) slides – presentation; (2) diagrams; (3) libraries – java and android; (4) data models; and (5) forms.

Participation Procedure

We defined the following steps for participation in the experiment:

1. The participants were divided into four groups randomly;

2. The researcher gave participants a set of documents containing UML diagrams, a dataset model and an interface message specification for each product, such as the material used in training session. Each participant was provided with a desktop computer with all requirements to develop a software product and an experimental form to register the time spent in the development process.
3. The participant read each given document;
4. The researcher explained the documents;
5. The participant read and clarified possible doubts about the products specifications;
6. Each participant received and used two randomly drawn methodologies for the development of a requested m-learning product. For each application, participants registered its duration (start time, end and breaks). At the end of the two development tasks they were asked to answer a feedback questionnaire, giving their opinion about the experimental execution and the technologies used.

3.4 Data Analysis and Interpretation

As the experiment session was finished, collected data was prepared in order to apply the statistical tests.

3.4.1 Descriptive Statistics

For each participant ("`Participant #`" column), we collected the following data: total time of implementation (t in minutes) and total number of faults (f), identified by testing procedures, and the mean calculation. These results are shown in Table 2 and the results for each participant are plotted in the box-plots of Figures 7 and 8.

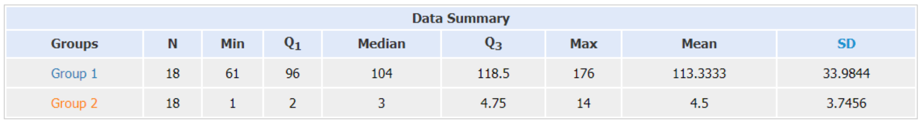
As we can see in Table 2 and in Figures 7 and 8, with regard to time-to-market, using SSD 50% of the developers took at least 104.00 minutes to develop an m-learning product with 33.98 minutes of standard deviation and maximum value of 176 minutes. On the other hand, using `M-SPLearning`, the 50% of the developers took at least 3 minutes with an standard deviation of 3.75 minutes and maximum time of 14 minutes. Therefore, the analyzed time-to-market between SSD and M-SPLearing is considerable different with better results to the `M-SPLearning` SPL.

Regarding the number of faults, using SSD, 50% of the developers found at least four faults with standard deviation of 8.46 and maximum value 34. By using `M-SPLearning`, 50% of the developers found no faults, with standard deviation of 2.91 and maximum value of nine faults. Thus, the analyzed number of faults favored `M-SPLearning` as it leads to a reduced number of faults during the m-learning product development.

Although, the results of analyzing the descriptive statistics of time-to-market and number of faults generally favored `M-SPLearning`, we conducted inferential statistics analysis to strength these results.

3.4.2 Inferential Statistics

Based on the results obtained by the use of SSD and SPL to the development of two m-learning products, we summarize, analyze and interpret the SSD and `M-SPLearning` collected data (Table 2 and Figures 7 and 8) by means of the Shapiro-Wilk normality test and the Mann-Whitney-Wilcoxon hypothesis test.



Data Summary								
Groups	N	Min	Q1	Median	Q3	Max	Mean	SD
Group 1	18	61	96	104	118.5	176	113.3333	33.9844
Group 2	18	1	2	3	4.75	14	4.5	3.7456

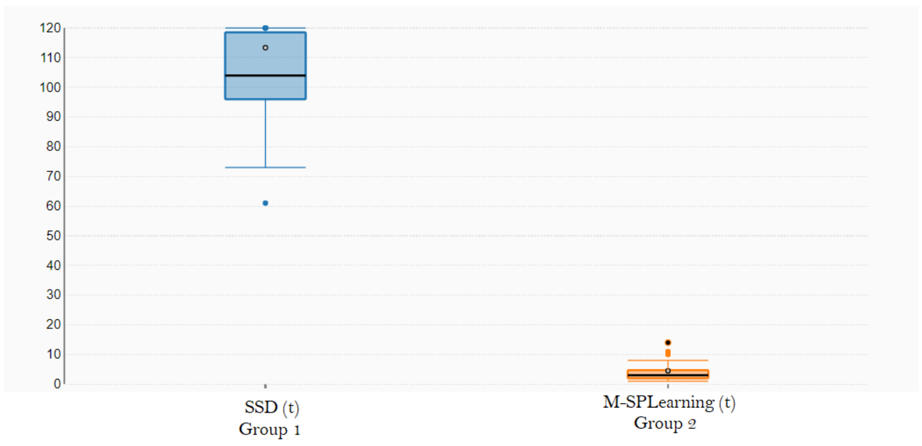
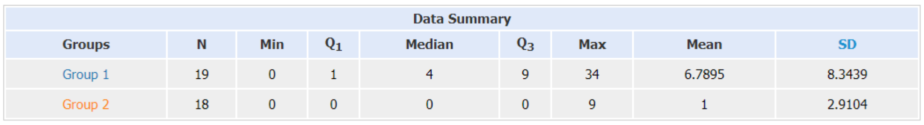


Figure 7: Boxplots for SSD and `M-SPLearning` time



Data Summary								
Groups	N	Min	Q1	Median	Q3	Max	Mean	SD
Group 1	19	0	1	4	9	34	6.7895	8.3439
Group 2	18	0	0	0	0	9	1	2.9104

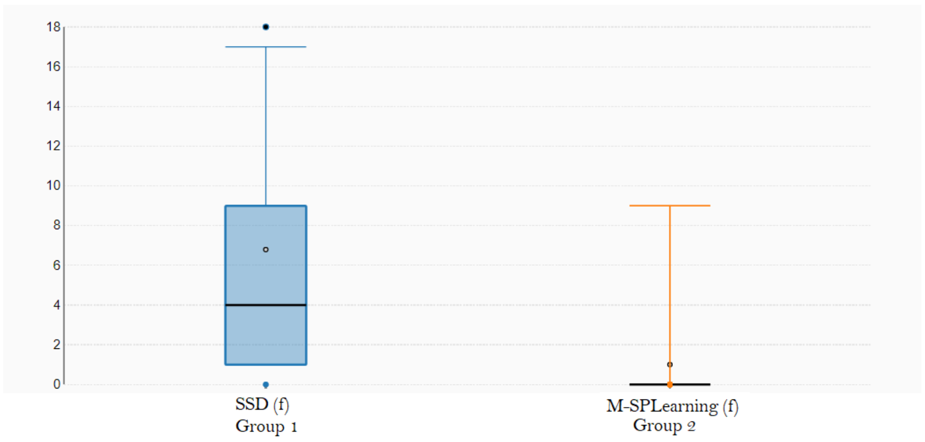


Figure 8: Boxplots for SSD and `M-SPLearning` faults

	SSD		M-SPLearning	
Participant #	**Time (t)**	**Faults (f)**	**Time (t)**	**Faults (t)**
1	161	15	2	9
2	90	8	1	0
3	105	4	11	0
4	104	1	3	0
5	73	2	1	0
6	99	9	3	0
7	165	12	10	0
8	95	1	3	0
9	104	3	2	0
10	102	0	4	0
11	61	0	2	0
12	82	4	8	0
13	114	1	4	0
14	103	6	14	0
15	111	2	2	0
16	176	9	4	0
17	120	17	5	0
18	175	34	2	9
Total	**2040 min**	**128**	**81 min**	**18**
Mean	**113.33 min**	**7.11**	**4.50 min**	**1**
Median	**104 min**	**4**	**3 min**	**0**
Std. Dev.	**33.98 min**	**8.46**	**3.75 min**	**2.91**

Table 2: SSD and SPL Collected Data and Descriptive Statistics.

Time-to-market (R.Q.1)

– **Collected Data Normality Tests:** The Shapiro-Wilk test [Shapiro and Wilk, 1965] was adopted for the analysis of the normal distribution of the samples. Being applied to the data set collected for the SSD and SPL methodologies in relation to the averages of time and faults, providing the following results for the time-to-market dependent variable:

- ***SSD time (N=18):***

  For mean value (μ) of 113.33 and standard deviation value of (σ) 33.98, the time for the SSD was p = 0.0274.

  In the *Shapiro-Wilk* test for a sample size *(N)* 18 with 95% of significance level ($\alpha = 0.05$), $p = 0.0274$ ($0.0274 < 0.05$) and calculated value of $W = 0.8813 < W = 0.8970$, the sample is considered non-normal.

- ***SPL time (N=18):***

  For mean value (μ) of 4.50 and standard deviation value of (σ) 3.75, the time for the SPL was p = 0.0014.

For a sample size *(N)* 18 with 95% of significance level (α = 0.05), *p* = 0.0014 (0.0014 $<$ 0.05) and calculated value of W = 0.7978 $<$ W = 0.8970, the sample is considered non-normal.

– **Mann-Whitney-Wilcoxon for SSD and SPL time samples:** a rank with weights was assigned to each sample value. The weights were added and applied in Equation 3:

$$U(DM) = N_1 * N_2 + \frac{N_1 * (N_1 + 1)}{2} - \sum_{i=1}^{n} total_2 \tag{3}$$

where:

$U(DM)$ equation for each independent sample (development methodology);

N_1 is the size of the sample for the X methodology;

N_2 is the size of the sample for the compared methodology (Y);

$total_2$ is the sum of the weights given for the compared methodology.

The time values calculated by Equation 3 were 326.5 for SSD and 0.00 for SPL. Each weight matches the participants weights of development process time with SSD or SPL methodology. Therefore, as both values are different ($326.5 > 0$), this leads to the rejection of the null hypothesis (H_0) and acceptance of the alternative hypothesis (H_1).

Based on the Figure 7 it is noticed that the time spend in SSD was higher, with the mean of 113 minutes while `M-SPLearning` took 4.5 minutes. However, it is important to highlight that such time does not include the time spent in the development of core of the products.

Therefore, the answer to R.Q.1 is that SPL is more efficient than the SSD to implement software products for mobile platform taking into account the products P1 and P2 specification. The implementations of the base project (used by SSD) and `M-SPLear ning` (used by SPL) were also considered in the experiment.

Regarding the base project, it consisted of a simple structure with only some abstractions and best practices of the company, also explored during the training sessions. This project was implemented in 480 minutes (8 hours), which is relevant for some analysis. In different circumstances, the `M-SPLearning` took 10599 minutes ($\approx$177 hours) to be developed, mainly due to the complexity of building/integrating core assets and generating products.

Based on the time spent in the development of each software product, it is possible to make some estimates related with the time efforts for both SSD and `M-SPLearning` methodologies evaluated. The total of time considering the time required for the implementation of each base project adopting the SSD methodology plus the total development time to each participant (480 minutes), the total time would be 10680 minutes or 178 hours ($total_{time}$((subjects(18) x minutes(480)) + 2040 = 10680 minutes).

On the other hand, taking into account the base project developing with SPL methodology, the time spent by the 18 participants was 81 minutes (1 hour and 21 minutes) and the total time would be 11361 minutes or 189 hours and 35 minutes ($total_{time}$((participants(18) x minutes(4.5)) + 10599 = 11361 minutes).

Comparing the values, we notice that the development with `M-SPLearning` takes 621 minutes (11 hours and 35 minutes) more than SSD. However, after the SPL implementation, this approach allows the evolution and insertion of new variabilities, assuring the faster generation of new products in addition to other advantages of the adoption of M-SPLearning. To conclude, according to [Linden et al., 2007], the Return On Investment (ROI) of an SPL usually occurs after the production of 3 to 4 products.

Number of Faults (R.Q.2)

- **Collected Data Normality Tests:** In this context, the same statistical tests applied to the time-to-market dependent variable were performed for the average of faults, starting with the Shapiro-Wilk test:
  - ***SSD faults (N=18):***

    For a mean value (μ) 7.11 and standard deviation value of (σ) 4, the fault for the SSD was p = 0.0006 for the *Shapiro-Wilk* normality test.

    For a sample size *(N)* 18 with 95% significance level ($\alpha = 0.05$), $p = 0.0006$ ($0.0006 < 0.05$) and value of $W = 0.7740 < W = 0.8970$, the sample is considered non-normal.
  - ***SPL faults (N=18):***

    For a mean value (μ) 1.00 and standard deviation value of (σ) 0, the fault for the SPL was p = 0.00000007 for the *Shapiro-Wilk* normality test.

    For a sample size *(N)* 18 with 95% of significance level ($\alpha = 0.05$), $p = 0.00000007$ ($0.00000007 < 0.05$) and value of $W = 0.3730 < W = 0.8970$, the sample is considered non-normal.
- **Mann-Whitney-Wilcoxon for SSD and SPL faults samples:** the number of faults calculated for SSD by Equation 3 was 282, whereas for M-SPLearning, it was 42. Each weight matches participants development project faults with SSD or M-SPLearning. There is evidence that both values are different ($282 > 42$), which leads to the rejection of the null hypothesis (H_0) and acceptance of the alternative hypothesis (H_1).

According to the result from the Mann-Whitney-Wilcoxon, the answer for R.Q.2 is that SSD is prone to present more faults in the software products developed than the M-SPLearning. Figure 8 confirmed such number, showing that SSD present a mean of 6 faults, while M-SPLearning had a mean of 1 fault.

The faults found were identified based on test cases defined with a group of quality analysts from a software industry encompassing both SPL and SSD methodologies. These test cases considered the main functionalists of the expected software products.

Table 3 summarizes our results in the normality and statistical tests. In terms of time-to-market, the statistical difference showed by Mann-Whitney-Wilcoxon test (MWW in Table 3) provides evidence that SPL was more efficient than SSD in the development of P1 and P2, so R.Q.1 was answered. Regarding the number of faults, the statistical difference presented by Mann-Whitney-Wilcoxon test provides evidence that SSD showed more faults than M-SPLearning in the development of P1 and P2; therefore, R.Q.2 has been answered.

Two types of faults were found for two participants that used the M-SPLearning, one for each configuration (P1 and P2). In both test cases, the product was expected to display a list of educational content (video or image), but nothing was displayed due to connectivity issues related to the corporate proxy.

Element	**SSD**	**M-SPLearning**
Participants	N(SSD) = 18	N(M-SPLearning) = 18
Time to Implementation		
Mean	113.33	4.5
Shapiro-Wilk	p = 0.0274 (p <0.05) Non-normal.	p = 0.0014 (p <0.05) Non-normal.
MWW	326.5	0
Result	R.Q.1 H2: M-SPLearning has a smaller time-to-market than SSD, based on evidenced statistical difference of the time to implementation.	
Number of Faults		
Mean	7.11	1
Shapiro-Wilk	p = 0.0006 (p <0.05) Non-normal.	p = 0.00000007 (p <0.05) Non-normal.
MWW	282	42
Result	R.Q.2 H_2: M-SPLearning has a smaller faults in the products than SSD, based on evidenced statistical difference of the number of faults.	

Table 3: SSD and `M-SPLearning` Normality and Statistical Tests Results.

For the SSD methodology, the faults were more serious. All the participants presented faults somehow. For the image (P1) and video (P2) software product, the faults where found in the same test cases: (i) the data was expected to be validated before be stored by the application to the data base in the content creation process; (ii) the application should present a form with the selected educational content, but nothing was displayed; and (iii) the informational messages were not displayed on some screens.

According to the results of the Wilcoxon test, both R.Q.1 and R.Q.2 null hypotheses can be rejected with a significance level of 95% ($\alpha = 0.05$). It is important to highlight that, even with a small number of participants and with a reduced statistical power, this experimental evaluation is important since it allows the collection of preliminary evidences about the compared methodologies. Besides, the sample can be increased in future replications [Falessi et al., 2018].

Additionally, it was also possible to identify clear differences between products screens (Figure 9). In this sense, `M-SPLearning` considered the best practices of the Android platform at the time. So an evaluation in terms of UI/UX could be conducted in the future.

3.5 Threats to Validity

This section addresses the actions taken to directly mitigate the threats of this experiment, according to the Conceptual Model of [Neto and Conte, 2013] and [Wohlin et al., 2012].

1. **Threats to Internal Validity:**
   - **Differences among participants:** as we selected participants with different experience levels, variations in their skills were reduced during the training

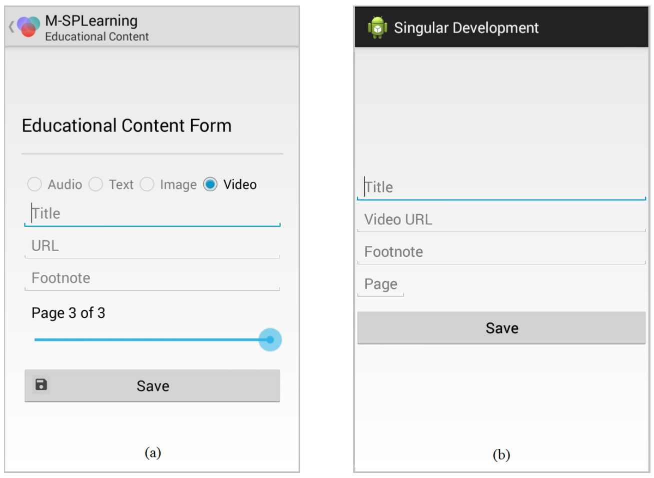


Figure 9: Example developed applications - P2: (a) `M-SPLearning` and (b) SSD.

sessions. The assessments conducted in the end of each day of training demonstrated the level of knowledge in the content used in the experimental execution and assured the reduction in variations in the participant skills. Even knowing that a more homogeneous sample reduces the participants representativeness, we decided to conduct a training session to reduce the heterogeneity of participants, that could threat the conclusion validity;

- **Fatigue effects:** on average, the experiment lasted 180 minutes. Fatigue was not considered relevant since the participants could leave the room for a quick break. They were warned to not communicate during the breaks and, as a guarantee, a human observer supervised them. Periods of absence were registered and disregarded in the time analysed;

- **Influence among participants:** the participants performed the experiment under the supervision of a human observer, then a possible influence of communication among them could be mitigated. Since participants behave differently when being observed, training sessions allowed the adaptation the participants to the environment, reducing this threat;

- **Training Sessions:** in the training sessions, technical explanations were given for every participant. This action was taken to avoid possible biases, in addition to encouraging each member of the training to expose their doubts.

2. **Threats to External Validity:**

   - **Instrumentation:** m-learning products and other instruments were tested in the pilot project and were considered significant for the analysis of time-to-market and number of faults;

   - **Participants:** more experiments considering different metrics with industry practitioners must be conducted for the identification of other relevant factors related to the adoption of `M-SPLearning`.

3. **Threats to Construct Validity:**
   - **Independent Variables:** independent variables were tested in the pilot project to guarantee their validity;
   - **Number of SPL products:** we are aware that SPL requires an initial investment to provide return on investment, e.g. the development of at least three products. However, as for this research context on m-learning applications there is no previous directly related work, we assume SPL and its inherited variability might provide insights on how it contributes to reuse core assets to produce prospective similar products. We discuss this in Section 4.
4. **Threats to Conclusion Validity:**
   - **Number of Participants:** since the number of participants was reduced, mainly by the availability of practitioners in the industry, the sample size must be increased in prospective replications of the experiment. Furthermore, according to [Falessi et al., 2018, Höst et al., 2000], the growth of experimental executions in an industrial environment, even with small samples, is important for the maturity of emerging development approaches;
   - **SPL Adoption Advantages:** it is known that not possible to cover the demonstration of all the benefits of the adoption of SPL based on just the creation of specific products of an SPL, since the core assets are fundamental. However, the analysis of development time of specific products and the quantity of faults are two metrics preconized by the software industry to guarantee a reduced time-to-market and costs of refactoring and maintenance. Finally, new experimental studies can be conducted, based on the experimental package of this study, allowing the investigation to consider more advantages or even disadvantages in the adoption of SPL.

4 Lessons Learned and Prospective Improvements

As the main lessons learned during the execution of the activities discussed herein, we highlight:

Domain Characteristics: domain analysis can be considered one of the most important activities for the creation of an SPL. The use of the requirements catalog [Duarte Filho and Barbosa, 2013] significantly contributed in terms of domain knowledge, supporting the adoption of the proactive model for the development of `M-SPLearning`.

Variability Management: the adoption of the SMarty approach supported the representation of variability points during the design of `M-SPLearning`, ensuring the synergy of the components modeled with the SPL concept. In addition, it contributed to a better assimilation of the SPL concept among the participants of the experiment, due to its compliance with the UML. Finally, we have already mentioned that SMarty has many publications validating its diagrams and, more recently, a model-based testing approach [Petry et al., 2021] has been proposed, which could be explored in future work.

SPL Architecture: SPL requires mechanisms to allow a transparent and easy manner to reflect all updates in the core assets in the architecture, and vice versa. That is, changes to the core assets require more efforts to maintain architectural integrity. Additional tools and analysis must be done to guarantee that all changes, or in the architecture or in the components, are reflecting all features and behaviors that the line has until that moment.

SPL Development: considering `M-SPLearning`, the implementation of something generic and customizable is significantly different from a SSD approach. Therefore, developing features in an SPL requires a greater effort, which is justified by the subsequent gains of reuse [Clements and Northrop, 2002].

On the other hand, the SSD methodology hinders the maintenance of the products, without any rigorous development process defined. Since all developers tend to program their own way, supporting each product they develop can, in most cases, take more time and costs. If an SPL practice is adopted, being rigorously followed, time and cost to give support tend to decrease.

Experimental Evaluation: researches show that test executions in SPL are scarce and need to be evaluated and validated [Engström and Runeson, 2011]. So, we decided to apply test cases in the generated products, enabling an interesting market comparison between development methodologies.

The experimental evaluation provided relevant results for the adoption of `M-SPLearning`. The choice for active participants from the industry contributed to the reduction of the training session. However, experience and understanding of the concepts is always a difficult issue to be measured.

In addition, the use of testing techniques in the traditional development process and new ways to quickly test products from an SPL require more attention and research. One benefit in the adoption of SPL is about the quality improvement, since the products and their components are tested in several instances, leading to a quickly fix for the final client. Despite the use of components in a large number of products and, consequently, for a greater number of target users for SPL, the way components and products were tested could be improved to be made more adequately with the variabilities of SPL. Literature reflects more concern to test the SPL architecture in comparison to test the final components and products [Machado et al., 2014, Petry et al., 2020].

5 Related Work

Our paper encompasses three main perspectives in industrial environments: (i) m-learning applications; (ii) SPL, with its benefits through variability management; and (iii) experimental software engineering. Adopting SPL as methodology to develop m-learning applications allowed us to get positive evidence in a real industry environment about two measurable SPL benefits – quality of products and time-to-market – when experimentally compared with a singular software development process without a variability management approach to support the developers. Thus, our results come to complement the experiences related in other researches for these three perspectives.

[Bezerra et al., 2009] conducted a systematic literature review of SPL applied to mobile middlewares, but only six studies were significant for the review. According to the authors, the few results obtained highlight the need of more research in the area. Besides that, [Chen and Babar, 2011], in other systematic review, concluded the status of evaluation of variability management approaches in SPL engineering was quite unsatisfactory. Thus, we identified some relevant related work, described in order of importance in the Table 4.

Author(s)	Domain	Findings
[Gamez et al., 2015]	SPL for mobile systems	Propose a self-adaptation of mobile systems with dynamic SPL. The management of variabilities is achieved using the Common Variability Language.
[Marinho et al., 2010]	SPL for mobile and context-aware applications	Describe an architecture for nested SPL in the domain of mobile and context-aware applications. However, the authors did not specify how to improve the management of variabilities.
[Jaring and Bosch, 2002]	SPL variability management	Conducts a case study focused on professional mobile communication infrastructures. They discuss the need for handling variability in a more explicit manner, analysing the SPL and a method to represent and normalize variabilities. Some issues were highlighted, such as the need of a notation format to describe the variabilities.
[Hubaux et al., 2011]	SPL variability management	Combine variability representation and industrial case studies evaluations. They developed a Textual Variability Language (TVL) combining graphical and textual notations and performed an evaluation through a quantitative and qualitative analysis, considering four cases from different companies, sizes and domains.
[Eriksson et al., 2009]	SPL variability management	Describes an approach to manage natural-language requirements specifications in an SPL context.
[Ardis et al., 2000]	SPL variability management and tests	Use the Family-Oriented Abstraction, Specification and Translation (FAST) approach as a development process for an SPL in a case study, covering all aspects of domain analysis with tests. According to the authors, test process in an SPL presents significant challenges.
[Gacek et al., 2001]	SPL tests	Presents a case study regarding the adoption of SPL in a small company. An holistic view of the challenges and changes in business was discussed, especially the automatization of tests in their developed components.

Table 4: Summary of Related Work

It is important to highlight that neither [Gamez et al., 2015] nor [Marinho et al., 2010] allowed us to conduct a direct comparison with respect to our work, because these works explore a more generic SPL domain. On the other hand, the works of [Hubaux et al., 2011], [Eriksson et al., 2009] and [Ardis et al., 2000] proposed the specification of variabilities through language and graphical representations in industry case studies. In our research, the representation of variability was made by using the SMarty approach, which has already been evaluated in other experimental studies [Marcolino et al., 2013, Marcolino et al., 2014b, Marcolino et al., 2014a, Bera et al., 2015, Marcolino et al., 2017].

Only two studies addressed some quality issues and tests, as our work. [Ardis et al., 2000] defined a testing approach based on modeling test cases with an integer tuple. However, the proposed method was applied only in the context of SPL, not considering the SSD methodology. [Gacek et al., 2001] adopted two-fold strategies in a case study to test the SPL products. Firstly, single components were tested by the developers themselves; secondly, the system was tested at runtime in the target environment, using special test code inserted in each component. Similar to the [Ardis et al., 2000] study, only the SPL methodology was considered, unlike our experiment, where quality analysts defined and executed test cases on products generated by two different development methodologies.

Finally, although all related works have been applied in the industry, none of them compare development methodologies, highlighting variables such as quality and time-to-market. Additionally, there are still several opportunities and open questions regarding systematic reuse and experimental evaluation, particularly in the m-learning domain. This lack of research in the area motivated our work.

6 Conclusions and Future Work

This work presented `M-SPLearning` – an SPL that intends to support the systematic generation of m-learning applications on the Android platform. According to [Linden et al., 2007], abstractions such as SPL allow the development process to be documented and reused systematically, contributing to the better comprehension of the target domain.

Considering the systematic studies carried out previously, there is a gap in the use of systematic reuse approaches (e.g., SPL) in the domain of m-learning applications [FalvoJr, 2015, Marcolino and Barbosa, 2015]. Highlighting the importance of research on approaches to reducing the software development workforce in this context; decreasing the time-to-market; increasing the quality of developed systems; among other benefits.

The main contribution of this work is related to the proposition of `M-SPLearning` and its experimental evaluation, which evidenced an improvement in the process of developing educational mobile applications through the use of SPL. Actually, the fault tolerance and time-to-market of such applications is essential to the end users. Therefore, there is a need to experimentally evaluate the products generated by the proposed SPL.

We experimentally evaluated the use of `M-SPLearning` with respect to the singular software development. The obtained results were significant for the reuse approach, showing a reduction on time-to-market and a better quality in terms of number of faults when considering the software products developed with the support of variabilities.

The SMarty approach was crucial to the design and development of `M-SPLearning`. The ease of importing the SMarty Profile into UML tools and the support provided by the SMarty Process in the representation of variability provided cost savings and better quality to the generated software products. In addition, with the experimental evaluations carried out, improvements could be applied to its elements, making SMarty a more complete and concise approach to use.

As important future work, we intend to evolve `M-SPLearning` based on the inputs provided by our experimental evaluation and considering GLO and LO support [Costea et al., 2021, Costea et al., 2018, Burbaite et al., 2014, Stuikys et al., 2013]. In this sense, new experimental evaluations can be planned from two perspectives: (i) replication of the experimental evaluation presented here, in order to increase the statistical power of the initial sample; and (ii) design and execution of experiments, aiming to evaluate the SPL itself and not just its respective products, as explored in this work. For this, there are currently well-defined guidelines to promote SPL experiments [Furtado et al., 2021], which can structure even more effective evaluations.

The conceptual model developed has also been evaluated and evolved, considering the specific domain of programming teaching. More studies on the adoption of `M-SPLear ning` in different contexts should result in improvements in our proposal, making it more suitable for the use in both academia and industry.

Moreover, for a complete evaluation of `M-SPLearning`, all features elicited by the catalog requirements must be properly implemented. Therefore, an experimental study involving the SPL itself should be conducted as well.

Finally, we also intend to investigate the use of other adoption models [Krueger, 2002]. 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.

Acknowledgements

The authors would like to thank CAPES/Brazil (PROCAD Grant number 071/2013) and FAPESP/Brazil (Grant number 2012/04053-9) for supporting this work.

References

[Anido-Rifon et al., 2001] Anido-Rifon, L., Fernández-Iglesias, M., Llamas-Nistal, M., Caeiro-Rodriguez, M., Santos-Gago, J., and Rodríguez-Estévez, J. (2001). A component model for standardized web-based education. *Journal on Educational Resources in Computing (JERIC)*, 1(2es).

[Ardis et al., 2000] Ardis, M., Daley, N., Hoffman, D., Siy, H., and Weiss, D. (2000). Software product lines: a case study. *Software: Practice and Experience*, 30(7):825–847.

[Bera et al., 2015] Bera, M. H. G., OliveiraJr, E., and Colanzi, T. E. (2015). Evidence-based smarty support for variability identification and representation in component models. In *Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 2*, ICEIS 2015, pages 295–302. SCITEPRESS - Science and Technology Publications, Lda.

[Bezerra et al., 2009] Bezerra, Y. M., Pereira, T. A. B., and da Silveira, G. E. (2009). A systematic review of software product lines applied to mobile middleware. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 1024–1029.

[Brooks et al., 1996] Brooks, A., Daly, J., Miller, J., Roper, M., and Wood, M. (1996). Replication of experimental results in software engineering. *International Software Engineering Research Network (ISERN) Technical Report ISERN-96-10, University of Strathclyde*, 2.

[Burbaite et al., 2014] Burbaite, R., Bespalova, K., Damasevicius, R., and Stuikys, V. (2014). Context aware generative learning objects for teaching computer science. *International Journal of Engineering Education*, 30(4):929–936.

[Capil la et al., 2013] Capil la, R., Bosch, J., and Kang, K.-C. (2013). *Systems and Software Variability Management: Concepts, Tools and Experiences*. Springer.

[Chau and Reith, 2021] Chau, M. and Reith, R. (2021). Smartphone market share.

[Chen and Babar, 2011] Chen, L. and Babar, M. A. (2011). A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, 53(4):344 – 362. Special section: Software Engineering track of the 24th Annual Symposium on Applied Computing.

[Clements and Northrop, 2002] Clements, P. and Northrop, L. (2002). *Software product lines: practices and patterns*. Addison-Wesley.

[Costea et al., 2021] Costea, F.-M., Chirila, C.-B., and Crețu, V.-I. (2021). Middle school arithmetic auto-generative learning objects to support learning in the covid-19 pandemic. In *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 000039–000044. IEEE.

[Costea et al., 2018] Costea, F.-M., Chirila, C.-B., and Crețu, V.-L. (2018). Towards auto-generative learning objects for industrial it services. In *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 000155–000160. IEEE.

[Craig and Jaskiel, 2002] Craig, R. D. and Jaskiel, S. P. (2002). *Systematic software testing*. Artech House.

[Dóra et al., 2013] Dóra, P. M., Oliveira, A. C., and Moura, J. A. B. (2013). Simultaneously improving quality and time-to-market in agile development. In *International Conference on Software Technologies*, pages 84–98. Springer.

[Duarte Filho and Barbosa, 2013] Duarte Filho, N. F. and Barbosa, E. F. (2013). A requirements catalog for mobile learning environments. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC)*, pages 1266–1271, New York, NY, USA.

[Engström and Runeson, 2011] Engström, E. and Runeson, P. (2011). Software product line testing – a systematic mapping study. *Information and Software Technology*, 53(1):2–13.

[Eriksson et al., 2009] Eriksson, M., Börstler, J., and Borg, K. (2009). Managing requirements specifications for product lines – an approach and industry case study. *Journal of Systems and Software*, 82(3):435 – 447.

[Falessi et al., 2018] Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., and Oivo, M. (2018). Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineering*, 23(1):452–489.

[FalvoJr, 2015] FalvoJr, V. (2015). Study and Definition of a Software Product Line for the Development of Mobile Learning Applications. Master's thesis, University of Sao Paulo (USP). In Portuguese.

[FalvoJr et al., 2014a] FalvoJr, V., Duarte Filho, N. F., OliveiraJr, E., and Barbosa, E. F. (2014a). A contribution to the adoption of software product lines in the development of mobile learning applications. In *Proceedings of the Frontiers in Education Conference (FIE)*.

[FalvoJr et al., 2014b] FalvoJr, V., Duarte Filho, N. F., OliveiraJr, E., and Barbosa, E. F. (2014b). Towards the establishment of a software product line for mobile learning applications. In *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 678 – 683.

[Furtado et al., 2021] Furtado, V., OliveiraJr, E., and Kalinowski, M. (2021). Guidelines for promoting software product line experiments. In *15th Brazilian Symposium on Software Components, Architectures, and Reuse*, SBCARS '21, page 31–40. Association for Computing Machinery.

[Gacek et al., 2001] Gacek, C., Knauber, P., Schmid, K., and Clements, P. (2001). Successful software product line development in a small organization. *IESE-Report No. 013.01/E.*

[Galster et al., 2014] Galster, M., Weyns, D., Tofan, D., Michalik, B., and Avgeriou, P. (2014). Variability in software systems: A systematic literature review. *IEEE Transactions on Software Engineering*, 40(3):282–306.

[Gamez et al., 2015] Gamez, N., Fuentes, L., and Troya, J. M. (2015). Creating self-adapting mobile systems with dynamic software product lines. *IEEE Software*, 32(2):105–112.

[Höst et al., 2000] Höst, M., Regnell, B., and Wohlin, C. (2000). Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214.

[Hubaux et al., 2011] Hubaux, A., Boucher, Q., Hartmann, H., Michel, R., and Heymans, P. (2011). Evaluating a textual feature modelling language: Four industrial case studies. In Malloy, B., Staab, S., and van den Brand, M., editors, *Software Language Engineering*, pages 337–356, Berlin, Heidelberg. Springer Berlin Heidelberg.

[ISO/IEC, 2010] ISO/IEC (2010). ISO/IEC 25010 System and software quality models. Technical report, International Organization for Standardization (ISO).

[Jabangwe et al., 2018] Jabangwe, R., Edison, H., and Duc, A. N. (2018). Software engineering process models for mobile app development: A systematic literature review. *Journal of Systems and Software*, 145:98 – 111.

[Jaring and Bosch, 2002] Jaring, M. and Bosch, J. (2002). Representing variability in software product lines: A case study. In *International Conference on Software Product Lines*, pages 15–36. Springer.

[Jedlitschka and Pfahl, 2005] Jedlitschka, A. and Pfahl, D. (2005). Reporting guidelines for controlled experiments in software engineering. In *International Symposium on Empirical Software Engineering*.

[Kang et al., 1990] Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study. Technical report, Software Engineering Institute, Carnegie Mellon University.

[Kraut, 2013] Kraut, R. (2013). Policy guidelines for mobile learning.

[Krueger, 2002] Krueger, C. W. (2002). Easing the transition to software mass customization. In *Proceedings of the 4th International Workshop on Software Product-Family Engineering*, pages 282–293, London, UK.

[Kukulska-Hulme and Traxler, 2005] Kukulska-Hulme, A. and Traxler, J. (2005). Mobile learning: a handbook for educators and trainers.

[Linden et al., 2007] Linden, F. J. v. d., Schmid, K., and Rommes, E. (2007). *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer.

[Machado et al., 2014] Machado, I., McGregor, J., Cavalcanti, Y., and de Almeida, E. (2014). On strategies for testing software product lines: A systematic literature review. *Information and Software Technology*, 56(10):1183–1199.

[Marcolino et al., 2014a] Marcolino, A., OliveiraJr, E., and Gimenes, I. (2014a). Towards the effectiveness of the smarty approach for variability management at sequence diagram level. In *Proceedings of the 16th International Conference on Enterprise Information Systems - Volume 2*, ICEIS 2014, pages 249–256. SCITEPRESS - Science and Technology Publications, Lda.

[Marcolino et al., 2014b] Marcolino, A., OliveiraJr, E., Gimenes, I., and Barbosa, E. (2014b). Empirically based evolution of a variability management approach at uml class level. *38th Annual International Computers, Software & Applications Conference*, 1:354–363.

[Marcolino et al., 2017] Marcolino, A., OliveiraJr, E., Gimenes, I., and Barbosa, E. (2017). Variability resolution and product configuration with smarty: An experimental study on uml class diagrams. *Journal of Computer Science*, 13:307–319.

[Marcolino et al., 2013] Marcolino, A., OliveiraJr, E., Gimenes, I., and Maldonado, J. C. (2013). Towards the Effectiveness of a Variability Management Approach at Use Case Level. In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 214–219, Boston, MA, US.

[Marcolino and Barbosa, 2015] Marcolino, A. S. and Barbosa, E. F. (2015). Software Product Lines in the Educational Domain: A Systematic Mapping. *Simpósio Brasileiro de Informática na Educação*, 1:239–249. In Portuguese.

[Marcolino and Barbosa, 2016] Marcolino, A. S. and Barbosa, E. F. (2016). Towards an M-learning Requirements Catalog for the Development of Educational Applications for the Teaching of Programming. *In: 2016 IEEE Frontiers in Education Conference (FIE), 2016, Erie (PA).*, pages 1–5.

[Marinho et al., 2010] Marinho, F., Costa, A., Lima, F., Neto, J., Filho, J., Rocha, L., Dantas, V., Andrade, R., Teixeira, E., and Werner, C. (2010). An architecture proposal for nested software product lines in the domain of mobile and context-aware applications. In *Proceedings of the 4th Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, pages 51–60, Salvador, BA, BR.

[Matzavela and Alepis, 2021] Matzavela, V. and Alepis, E. (2021). M-learning in the covid-19 era: physical vs digital class. *Education and Information Technologies*.

[Moreira and Rocha, 2018] Moreira, F. and Rocha, Á. (2018). A special issue on disruption of higher education in the 21st century due to icts.

[Neto and Conte, 2013] Neto, A. A. and Conte, T. (2013). A conceptual model to address threats to validity in controlled experiments. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, EASE '13, pages 82–85, New York, NY, USA. ACM.

[OliveiraJr et al., 2010] OliveiraJr, E., Gimenes, I. M. S., and Maldonado, J. C. (2010). Systematic management of variability in uml-based software product lines. *Journal of Universal Computer Science*, 16(17):2374–2393.

[Petry et al., 2021] Petry, K., OliveiraJr, E., Costa, L., Zanin, A., and Zorzo, A. (2021). Smartytesting: A model-based testing approach for deriving software product line test sequences. In *Proceedings of the 23rd International Conference on Enterprise Information Systems*, pages 165–172. INSTICC, SciTePress.

[Petry et al., 2020] Petry, K. L., OliveiraJr, E., and Zorzo, A. F. (2020). Model-based testing of software product lines: Mapping study and research roadmap. *Journal of Systems and Software*, 167:110608.

[Shapiro and Wilk, 1965] Shapiro, S. S. and Wilk, M. B. (1965). An Analysis of Variance Test for Normality (Complete Samples). *Biometrika*, 52:591–611.

[Sharples, 2013] Sharples, M. (2013). Mobile learning: research, practice and challenges. *Distance Education in China*, 3(5):5–11.

[Sharples et al., 2009] Sharples, M., Arnedillo-Sánchez, I., Milrad, M., and Vavoula, G. (2009). Mobile learning. In *Technology-enhanced learning*, pages 233–249. Springer.

[StatCounter, 2021] StatCounter (2021). Mobile os market share worldwide.

[Stuikys et al., 2013] Stuikys, V., Burbaite, R., and Damasevicius, R. (2013). Teaching of computer science topics using meta-programming-based glos and lego robots. *Informatics in Education*, 12(1):125–142.

[Wohlin et al., 2012] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer Science & Business Media.