From e-Gov Web SPL to e-Gov Mobile SPL

Camilo Carromeu
Embrapa Beef Cattle, Brazilian Agricultural Research Corporation, Campo Grande, Brazil, and
Debora Barroso Paiva and Maria Istela Cagnin
College of Computing, Federal University of Mato Grosso do Sul, Campo Grande, Brazil

Received 22 October 2015
Accepted 5 November 2015

Abstract

Purpose – This paper aims to discuss the motivation and present the evolution from a Software Product Line (SPL) in the e-Gov Web (e-Gov Web SPL) domain to a SPL in the mobile domain (e-Gov Mobile SPL).

Design/methodology/approach – The evolution was supported by the Product Line UML-Based Software Engineering approach and the feature model.

Findings – The authors were able to observe that it is feasible to evolve from a SPL for the Web platform to a SPL for the mobile platform, with the intent to port existing Web applications to mobile platforms such that users can have access to the main information and are able to interact with the most important functionalities of Web applications in a mobile device.

Research limitations/implications – As for the main limitations, the authors can point out the small number of instantiations performed until the moment with the support of the e-Gov Mobile SPL, what prevented the conduction of an empirical study.

Practical implications – Using e-Gov Mobile SPL, it is possible to reduce development time and cost.

Originality/value – The existing SPLs do not worry about supporting the development of mobile applications corresponding to existing Web applications, as it is desirable to have access to the information and main features of these applications in mobile devices. We obtained some e-Gov Mobile SPL instantiations corresponding to e-Gov Web SPL instantiations to attend the demands of the Brazilian Agricultural Research Corporation Unit situated at Campo Grande, MS, Brazil.

Keywords Mobile, Web, Evolution, Software product line, E-Gov

Paper type Research paper

1. Introduction

Many studies had been done in the past years about Software Product Lines (SPLs) and its applications on different platforms (such as Web and mobile devices) and domains (such as education and games). These studies generally present evolutions and

This work is supported by FUNDECT (Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul – Brazil). This work is an enhanced version of the publication "The Evolution from a Web SPL of the E-Gov Domain to the Mobile Paradigm" by C. Carromeu, D.M.B. Paiva, and M.I. Cagnin, in Lecture Notes in Computer Science, Volume 9155, 2015, pp. 217-231 from The 15th International Conference on Computational Science and Its Application (ICCSA' 2015).

International Journal of Web Information Systems
Vol. 12 No. 1, 2016
pp. 39-61

1744-0084
DOI 10.1108/IJWIS-10-2015-0036

adaptations to the original concepts of the SPL to fit specific situations. It can be mentioned as an example the demand to develop mobile applications with functionalities that match Web applications with intent to allow the creation of applications that are executed in different environments and platforms. Thus, the portability non-functional requirement that was defined in international standards (ISO/IEC 9126:2001, 2001; ISO/IEC 25010:2011, 2011) and which refers to a set of sub-characteristics, such as adaptability and replaceability, can be satisfied.

In 2010, an SPL was developed for the e-Gov Web (e-Gov Web SPL) domain (Carromeu *et al.*, 2010) aiming to obtain a flexible architecture that could reuse components developed from several previous experiences from the research group LEDES (Laboratory of Software Development from UFMS) and the PLEASE Lab (Laboratory for Precision Livestock, Environment and Software Engineering from Embrapa). This architecture made possible the development of several e-Gov Web applications which were used in different real-life situations such as the Pandora (Brazilian National Institute of Industrial Property number: 012120000833), a Web application which makes available to the user information and services from the information and management systems from the Embrapa Beef Cattle.

With the availability of mobile devices, such as cellular devices and tablets, we observed that the applications developed based on the mentioned architecture needed to progress and adapt themselves to new platforms, operational systems and mobile technologies. Thereby, mobile applications were developed corresponding to the Web applications, but without the e-Gov Web SPL support. From these mobile applications, we observed that a common set of services was implemented in their majority. Then, the e-Gov Web SPL could evolve to a SPL for the mobile platform. To accomplish that, the common services in the developed mobile applications were abstracted as features (that are end-user visible characteristics of a system) (Kang *et al.*, 1990) and originated the e-Gov Mobile SPL (Carromeu *et al.*, 2015), which is an evolution of the e-Gov Web SPL. The features were represented in a feature model (Kang *et al.*, 1990) which, according to Sayyad *et al.* (2013), allows visualization, reasoning and configuration of SPLs. It can consist of hundreds (even thousands) of features with complex dependencies and constraints that govern which features can or cannot live and interact with other features.

This paper aims to describe the evolutions which were made in the e-Gov Web SPL to obtain the e-Gov Mobile SPL having in mind the need to attend a new market demand which is to make available Web applications in mobile environments, as there is a lack of works that contemplate this demand (Nascimento *et al.*, 2008; Quinton *et al.*, 2011; Marinho *et al.*, 2013; Mizouni *et al.*, 2014; Falvo Junior *et al.*, 2014; Pascual *et al.*, 2015). The results of an instantiation and a number of case studies of the e-Gov Mobile SPL are also presented aiming to show the application generation capacity for both Web platform and mobile platform.

This paper is organized as follows. Section 2 presents the related works. Section 3 describes the background to ease the understanding of concepts treated in the paper. Section 4 presents a brief description of the e-Gov Web SPL. Section 5 presents the evolution of the e-Gov Web SPL for the conception of the e-Gov Mobile SPL, as well as one resulting instance. Section 6 shows some instances of the e-Gov SPL Mobile, resulting from case studies. Section 7 debates contributions, limitations and suggestions for future works.

2. Related work

There are some works about mobile SPLs in the literature and all of them use the feature model for the variability modeling. As the goal of the studied works is to develop an application in the mobile platform with the help of a SPL, all of them treated the variability of the business domain (also called vertical domain) and the variability of mobile domain (which can be considered a horizontal domain for it can be present in different vertical domains as in the case of context awareness, location, battery level, security, communication, etc.).

Nascimento *et al.* (2008) present an approach to implement core assets in a SPL applied to the mobile game domain combining good practices of a process. They also present a case study performed with the application of the approach based on three different adventure mobile games. Results have shown that the approach can be suitable for the domain.

Quinton *et al.* (2011) propose a SPL approach based on Model Driven Engineering, whose objective is to elaborate variability modeling with the support of the feature models of both system's business domain and the mobile devices domain to derive a software product taking into consideration both models. For this, the authors present a derivation process supported by a framework that performs the combination of both models (model merging) and the code generation. Additionally, the derivation process automatically analyzes restrictions for each derived product in relation to the available mobile devices.

Marinho *et al.* (2013) present the SPL MobiLine, which supports the development of tour guide context-aware mobile applications, that is, the software must be capable of adapting itself, during runtime, according to changes in its context (for example, user location and network conditions). The authors used the features model to document both tour guide domain-related features and context-aware mobility-related features. In another study, Marinho *et al.* (2011) propose a mechanism to formalize and verify the correctness and consistency of feature models for mobile and context-aware SPLs.

Mizouni *et al.* (2014) propose a framework to build adaptive context aware and mobile applications, considering that context information can be related to the environment, users or device status. This framework is based on both variability modeling, with the support of the feature model and SPL. The adaptability modeling is done during the design phase, and the adaptability and context awareness are supported in execution time.

In another context, Falvo Junior *et al.* (2014) discussed the establishment of M-SPLearning, a SPL to the mobile learning applications domain. It has been developed throughout a proactive adoption model, according to the basis of Service-Oriented Architecture. Results suggest the practical feasibility of adopting M-SPLearning in the development of mobile learning applications.

Pascual *et al.* (2015) consider the Dynamic Software Product Lines (DSPL) for mobile devices. According to authors, it is a well-accepted approach to manage runtime variability, by means of late binding the variation points at runtime. In this study, evolutionary algorithms were used to generate at runtime optimum configurations of the DSPL according to different criteria. The optimization problem is formalized in terms of a feature model.

However, none of the studied SPLs considers the portability of the developed software system, as is the goal of the SPL approached in this work, that concerns about

the development of a mobile application corresponding to a Web application and with the communication between them as well. That makes possible for the user to have access to the main information and to interact with the most important functionalities of the Web application in a mobile device.

3. Background

SPL is one of the existing software reuse techniques and is of interest for this work because it provides the reuse of software products in the domain of e-Gov Mobile applications corresponding to e-Gov Web applications. The SPL was conceived from the adaptation of concepts of the Product Line Engineering, which is commonly applied in the engineering to reach mass customization with economy of efforts through the collective production (instead of individual) of multiple similar instances, but from distinct design and prototypes (Weiss and Lai, 1999).

According to Clements and Northrop (2002), an SPL is an set of intensive software systems that shares a set of managed and common features that meets the specific needs of a market segment in particular (domain) or company mission and is developed from a core set of assets in a pre-established way.

In a specific business domain, there are the mandatory characteristics (commonalities) and those that can be customized (variabilities). Both can be represented in variability models such as the feature model. According to Pohl *et al.* (2005), a variability is composed by a variation point and its variants. A variation point is the place where the variation may occur (payment methods, for example) and the variants are the existing possible solutions for a variation point (for example, money, credit card or banking billet). Thus, the management of variabilities aims to organize commonalities and variabilities in a way to generate products with higher quality, reducing the use of organizational resources.

Still according to Pohl *et al.* (2005), a SPL is built and instantiated from two processes named, respectively, Domain Engineering and Application Engineering. Basically, the Domain Engineering focuses on the variability management of all software artifacts from the SPL and Application Engineering is responsible for creating the final product from the SPL. In this work, the software artifacts from the presented e-Gov Mobile SPL are implemented with the support of application frameworks (Foote and Johnson, 1988; Sommerville, 2010) and application generators (Cleaveland, 1988), whose definitions are presented as follows. These mechanisms together provide the instantiation of the e-Gov Mobile SPL without spending much effort by the application engineer.

The application frameworks are considered as semi-complete and reusable applications that, when specialized, produce personalized applications within a specific domain (Foote and Johnson, 1988). They are composed by a collection of abstract and concrete classes, and interfaces between them, representing the frozen spots and hot spots of the project of a subsystem (Sommerville, 2010).

Application generators are software systems that transform the specifications into an application. The specifications describe the problem or the task that the application has to perform, such that the generator can generate the source code. These specifications can be modeled in a graphic way, written in some intermediate language or even be created interactively, where the user selects desired features through the choices in a sequence of forms or menus (Cleaveland, 1988).

4. SPL for e-Gov system development

With the objective of reuse in different levels of abstraction, software patterns appeared in the 90s (Buschmann *et al.*, 1996; Coplien, 1998, Gamma *et al.*, 1995) to try to capture acquired experience in the software development and synthesize it in the form of problem and solution. Besides providing the reuse of the solutions, patterns help to improve the communication among developers, which can conduct the discussions based on the usage of these patterns (Gamma *et al.*, 1995).

In this context, several experiences in the LEDES and in the PLEASE Lab in development of Web applications in the e-Gov domain resulted in the abstraction of analysis, architecture and codification patterns. Aiming to consolidate an agile development process for this domain, we decided to create a SPL, named e-Gov Web SPL.

Considering the abstracted patterns from different members of the SPL from e-Gov domain, the e-Gov feature model was created and the initial architecture of the e-Gov Web SPL was established, whose development was based in the Product Line UML-Based Software Engineering (PLUS) approach (Gomma, 2005). This approach is composed of two processes: Software Product Line Engineering and Application Engineering.

In the Software Product Line Engineering, the commonalities and variabilities shown in the SPL feature model were mapped for the Titan Framework through classes, specifying the frozen-spots and the hot-spots, respectively, aiming to ease the instantiation of the line during the Application Engineering Process. The e-Gov Web SPL is automatized by the Titan Framework and the application generator Titan Architect, as detailed by Carromeu *et al.* (2010).

The use of the Titan Architect eliminates the need for programming, although the parametrization of variabilities is limited. The Titan Framework allows the configuration of business rules and parameters of the new application by modifying the markup language input and, in case it is not enough, it allows the programming of new software artifacts.

The Titan Framework architecture, illustrated in Figure 1, is formed by a core and by the repository of components. The core is the implementation of all SPL similarities and, hence, it is unchangeable in the line instances. It is responsible for receiving as input the instance configuration files (XML and SQL) and generating an application in runtime. This framework is a gray box, therefore, flexible and extensible.

The main features of the e-Gov Web SPL are the navigation sections and navigation actions which are related, respectively, to the components and engines. The role-based access control, the audit log record, the search engine, the file system, the notification system, the manual generator and the backup system on demand are other features inherent to its architecture.

5. Evolution of SPL for the mobile computing paradigm

Since the creation of the e-Gov Web SPL, dozens of new Web applications were developed and others are in development at the moment. At the same time, the increasing use of mobile devices and the wide dissemination of wireless networks are increasingly stimulating the development of softwares in the mobile and ubiquitous computing paradigms. In this context, the e-Gov domain is one of the main application

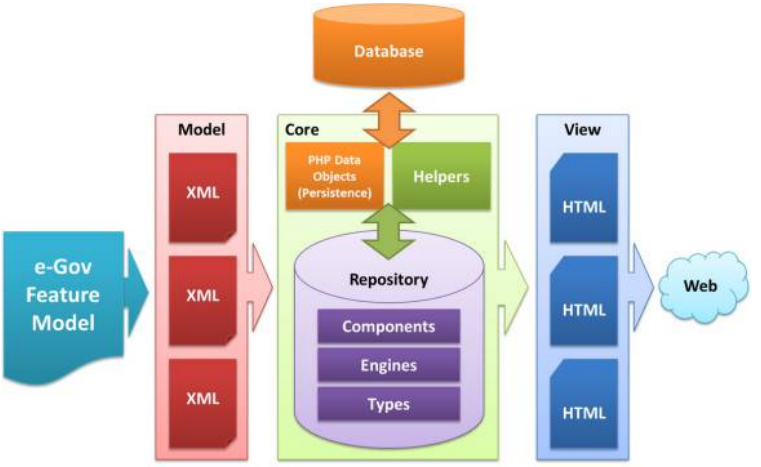


Figure 1.
Titan framework architecture

Source: Carromeu *et al*. (2010)

areas, once governments have been pushed to present higher efficiency when providing information and services to the society in a transparent and democratic way.

This way, it was identified a strong demand for mobile applications integrated with the developed Web systems, enabling the user access to the main information and functionalities of the system. For example, Suplementa Certo (Brazilian National Institute of Industrial Property number: 5120130013755), a calculator that helps with decision-making about feeding supplementation for beef cattle; and the Pandora Phone, that enables the access, by mobile devices, to some of the functionalities of the Pandora Web application, such as notifications, technology and knowledge bases, information about projects budgets (balance and statements) and the staff (employees and collaborators).

The experience with the development of several applications in the mobile platform, such as the mentioned above, enabled to initially identify codification patterns. For example, the code structure, in native Java, obeyed a recurring logic organization and demanded the same set of auxiliary classes (helpers).

Similarly, the implementation of the developed mobile applications required alterations in the Web applications (e-Gov Web SPL instances) with which they would have to synchronize data. To do that, it was necessary to implement exclusive service buses, related to communication and security and decoupled from the repository components from the SPL as discussed below.

5.1 Discussion of bus services implemented in the mobile applications

One of the main services implemented was the communication, which is fundamental to keep the mobile application data consistent with the Web application. Furthermore, a mobile device is more exposed to risks than a personal computer, and the data synchronization with a remote server is one way to prevent eventual losses. This communication has to be made through Web services, being also necessary to worry about security, performance and data integrity between the server and several clients.

Aiming to mitigate risks associated with *security* through *authentication* in the communication between client (mobile application) and server (Web application), it was specified a global scheme of authentication which can be used on any service available under the HTTP protocol, named Embrapa-Auth. It is a protocol that defines a scheme sufficiently secure, flexible, standardized and stateless to authenticate HTTP requests. The protocol uses headers and status codes from the HTTP protocol, favoring its usage in Representational State Transfer (REST) approaches. The protocol was developed to support an architecture where there are several users making HTTP requests to a Web service through applications running in several devices and platforms. The protocol is adaptable to the context, allowing up to three levels of authentication:

- *Application authentication*: An authorized application is any program that has an identifier/token pair that authorizes it to make requests to the service. The credentials of the application have to, preferably, be embedded to the source code of the program and should not be editable by an ordinary user. This level of security allows to restrict that only some specific programs are able to make requests to the service.
- *Client authentication*: A client is a device that has a private identifier/key pair which authorizes it to make requests to a service. The authentication of a client allows to restrict the use of Web services by determined devices or specific systems. This security level allows implementations of quota policies; restriction of functionalities; sharing of keys by company, unity, department, team, etc.; authorization of third-party softwares; auditing; change of private key in case equipment loss or theft; cancellation of credentials at the end of contracts; and user identification by a personal device, etc.
- *User authentication*: A user is somebody who owns an access credential (login/password pair).

Another implemented service refers to the *user registration* and *authentication of user by social network*. In the case of mobile devices with the Android operating system, it is possible to state that the application user has, necessarily, a Google account. This fact enables the user to use the account that is registered in the mobile device to make his/her registration in the mobile application. The biggest advantage of this approach is to offer the user a simplified and quick way to register without the requirement of filling out forms.

Another implemented service was to guarantee the local *data integrity* of mobile application in relation to the Web application.

It is easy to guarantee the data integrity when there is no creation of tuples in the mobile application database, as in the case of metadata entities (federative unit, country, marital status, etc.). In those cases, the mobile application assumes an architecture "*provider-consumer*", where the Web application will be the data provider and the mobile application will just "consume them". This way, the data synchronization controller shall be responsible for consulting the Web application and obtain new and updated tuples, so the data are synchronized from the Web application and cannot be locally modified by the mobile application.

However, often the mobile application must be responsible for the creation and modification of data, which will, at some point, be synchronized with the Web

application data. In this case, it is necessary to worry about the fact that the same data can be modified in different devices and, when synchronizing with the Web application, it is necessary to *handle conflicts*. The adopted policy in this case is to always keep the most recent data.

The *disambiguation* service was implemented to solve the problem when tuples of entities created in mobile applications are sent to the Web application and synchronized in the mobile applications installed in other devices. In this case, there is no way to guarantee that it will be possible to consult the Web application to obtain a unique key, as, in many cases, the mobile application will be able to work without internet access. It is difficult, in this case, to guarantee that the sequential key be correctly interpreted by the mobile application, as there will be other mobile devices creating tuples in the same entity.

Through the disambiguation service, the Web application provides a unique key to the mobile application once it is installed, where it is stored locally on the device. This key is always concatenated with the entity key when the latter is a sequential key.

The *notification* service was also implemented to allow notifications launched by the Web application to be reflected in the mobile application. For this, services were created in the service bus that interact with the notification API from the Titan Framework and allow:

- to access the message list and related data (date, category and link);
- to mark a notification as "read"; and
- to delete a notification.

Moreover, the notifications API from Titan Framework were integrated with the notifications API from Android, named Google Cloud Messaging. This way, alerts issued on the Web application are received by mobile application users in a passive way, that is, without the need of the mobile application to be running in that moment.

Finally, the **image-processing** service was also considered because mobile applications that try to make the processing of big images locally are very susceptible to errors related to the lack of memory. Through this service, images loaded from the Web application are remotely processed and simply put in cache in the mobile application.

5.2 E-Gov Mobile SPL conception

The implementation of the services presented in the previous section motivated the e-Gov Web SPL evolution to become an e-Gov Mobile SPL, being this evolution the focus of this work.

The e-Gov SPL Mobile features were identified from the services presented in Section 5.1 and are represented in the feature model illustrated in Figure 2. Next, for the incorporation of the raised features in the e-Gov Mobile SPL, there were used four evolution cycles of the Software Products Line Engineering process from the PLUS approach. This approach was used in the evolution of the e-Gov Web SPL to an e-Gov Mobile SPL, as, beyond having being used in the creation of the e-Gov Web SPL, demonstrated to be effective during the held evolution.

Consequently, the execution of the evolution cycles are in compliance with the execution description of the PLUS approach commented in Section 4, that is, the features presented in Figure 2 were mapped to the Titan Framework as frozen-spots or hot-spots;

and existing components in its repository were refactored to be able to interact with the incorporated services, as illustrated in Figure 3.

The first evolution cycle of the e-Gov Web SPL started with the modification of its fixed core, that is, the artifacts which materialize communalities from the members of the family of products. In this core, there was added the service bus with the basic services represented by the feature *AuthenticationLayer*. This parameterizable implementation of the Embrapa-Auth protocol guarantees that any SPL instance can count on any combination of the authentication levels of this protocol represented in the feature model from Figure 2.

The feature *ProviderConsumer* was implemented to guarantee that all instances from the e-Gov SPL Mobile will have entities that will be treated in a provider–consumer architecture. A problem that was observed during the implementation of this feature was the overhead of the initial synchronization, that is, when the mobile application is installed. This can become a problem when there are millions of tuples that must be

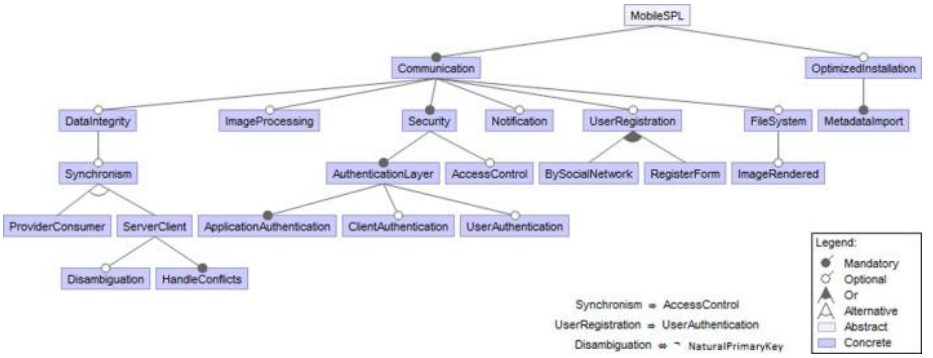


Figure 2.
Feature model of the evolution from an e-Gov Web SPL to an e-Gov Mobile SPL

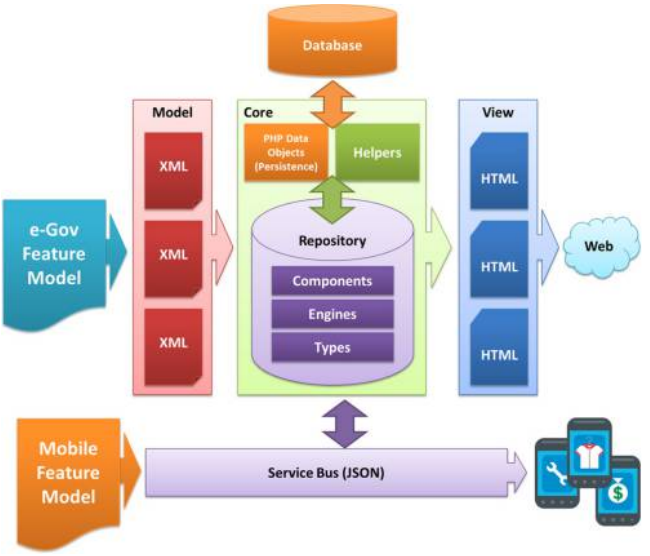


Figure 3.
Evolved Titan Framework Architecture for the development of mobile applications

synchronized. To optimize the initial synchronization, this feature was implemented taking into consideration the sending of a binary database with all the metadata stored, that is, when the mobile application is installed in the mobile device, the installer already has a binary database with metadata tables.

Additionally, in this first evolution cycle, the Titan Framework has acquired its service bus, but lacks the corresponding maturity. The repository components were adapted to interact with the new layer of communication (*Communication* feature). For example, the classes that implement data types from the Titan Framework acquired the capacity to represent and understand values in the JavaScript Object Notation (JSON) format, which is a format more compact than XML and thus chosen for the transmission of data between client and server.

It was also implemented, in this first cycle, the authentication layer (represented by the feature *AuthenticationLayer*) and its levels, represented by the features *ApplicationAuthentication*, *ClientAuthentication* and *UserAuthentication*. In the end, the service bus from Titan Framework was also capable to synchronize data in architectures provider–consumer and client–server and to treat inherent conflicts.

After the above-mentioned adaptations, it became evident the need for new improvements in the synchronization process, because of the already discussed problem, about the creation of tuples in entities without natural primary keys. Then, in the second evolution cycle, it implemented the *Disambiguation* feature. This evolution required refactoring of classes from the Titan Framework that implement the model and control layers of the Web application, aiming to add to the artifacts of these layers, the capacity to deal with the disambiguation code. Another implemented feature in this cycle was the *ImageProcessing*, that is, the services were made available in the service bus to allow the mobile application to request the server for already processed images.

In the third evolution cycle, it was made the integration of the Titan Framework with the Google's APIs for Android. This integration allowed the user registration to be made in a simple and transparent way, using personal information already available in the device (*BySocialNetwork* feature). It was also added the previously discussed integration of the notification API from Titan Framework with the Google Cloud Messaging (Section 5.1).

Together with the implementation of all raised features corresponding to the service bus (Section 5.1), with the support of the PLUS approach, all the new e-Gov SPL instances are apt to use them, in other words, in each of them, the application engineer can activate the service bus with different combinations of features, according to the rules and restrictions detailed in the feature model presented in Figure 2.

To facilitate the e-Gov SPL Mobile instantiation, it was created a code generator in the fourth and last evolution cycle, named Titan Architect Mobile, which allows the application engineer to generate the mobile application code. For this, this generator scans the files in markup language (XML) from the Web application, responsible for the parametrization of the Titan Framework components, generating a Java code that composes the mobile application project. The basis of this project is also fixed and consists of a structure of directories according to the Android pattern and a collection of auxiliary classes, named helpers, which implement functionalities common to all members of the product line.

Based on the files that parameterize the Web application, the Titan Architect Mobile generates the following classes for each mobile application entity, corresponding to the Web application entities explicitly mapped (in a proper XML file) for the mobile platform:

- *Contracts*: Classes responsible for the definition of the entity tables in the relational database of the mobile application. These classes also "know" how to create these tables charts when the application is installed.
- *Models*: Classes that define the data model of the entity. When the code is generated, a transcription of the instance's XMLs is done (these XML files containing each entity attribute and its respective data type) for the creation of these classes in the mobile application.
- *Data access objects (DAOs)*: Classes that implement the persistence layer, making the interaction with the database.
- *Adapters*: Classes with methods to convert values from the entities attributes. They are essential in the representation of classes objects as JSONobjects and for the conversion of values for the data record and recovery in the mobile application database.
- *Tasks*: Classes whose instances are called asynchronously by the mobile application and are responsible by the synchronization of the entity data with the Web application.
- *Web services*: They are classes that convert and send entities to the Web application and recover their information. They know the REST communication protocol and know how to treat errors when needed.

Moreover, the Titan Architect Mobile generates artifacts for the visualization layer (which includes classes called Views in the Android and XML files) in a visual pattern that can be easily altered by the application engineer. Optionally, a binary database is created (corresponding to the *MetadataImport* feature) and it is loaded when the mobile application is installed, as previously mentioned.

The code generated by the Titan Architect Mobile interacts, by means of message exchange, with a set of artifacts that implement parts of the application corresponding to the frozen spots from the Titan Framework. These artifacts are classes whose code is not modified from one instance to another. Make part of this static code diverse libraries, such as the ActionBarSherlock (2012) and the SlidingMenu (2014), and also common use classes made available in a package named "util". Examples of entities from this package include the classes CryptHelper, Database, DrawableHelper, EmbrapaAuthCredential, LogHelper, NetWorkVerifyer and ScreenHelper, whose names are self-explanatory.

Interventions in the source code obtained can be made by the application engineer to finish it and customize it, but not to optimize its performance. We highlight that the source code generated by Titan Architect Mobile will likely be more efficient than the source code manually implemented by a Java developer, who ignores or disregards the best practices of Android (2015) development. We can say this because the implementation of the generator involved detailed study of these "best practices". Thus, as the source code generation of Titan Architect Mobile considers these assumptions, then we can infer that the generated source code will be efficient.

5.3 Instantiating the e-Gov Mobile SPL

As shown in Figure 4, the instantiation of the e-Gov Mobile SPL happens from a Web instance of the Titan Framework. An instance from the Titan is formed by components of the e-Gov Web SPL that were parameterized to generate, at runtime, the functionalities of the Web application allocated in the business layer. Business rules and data models specified, respectively, in the Model Layer Specification and in the Persistence Layer (illustrated on Figure 4) for the instantiation of the e-Gov Web SPL are reused in the e-Gov Mobile SPL for the code generation of the mobile application.

The application engineer can choose which functionalities from the Web application will be present in the corresponding mobile application. For this, he or she must specify the functionalities in a XML markup file that has a similar syntax to the shown in Figure 5. It is necessary to specify in this file, the name of the mobile application ("Travel", in the example) and the entities from the Web applications that will be present in the mobile application ("Embassy", in the example).

Based on the aforementioned specification, the Titan Architect Mobile traverses, in the Web application, the artifacts that compose the mapped functionalities. In this case, for each functionality, the definition files from the models that compose its entities and the business rules contained in the component reused from the Titan Framework are considered to obtain the desired functionality. This way, the parametrization of the component that was instantiated to obtain the section in the Web application (in the example, section = "embassy") is reused for the generation of the mobile application. The remaining attributes from the XML specification (Figure 5) complete the information needed for the generation of the code for the mobile application, such as the

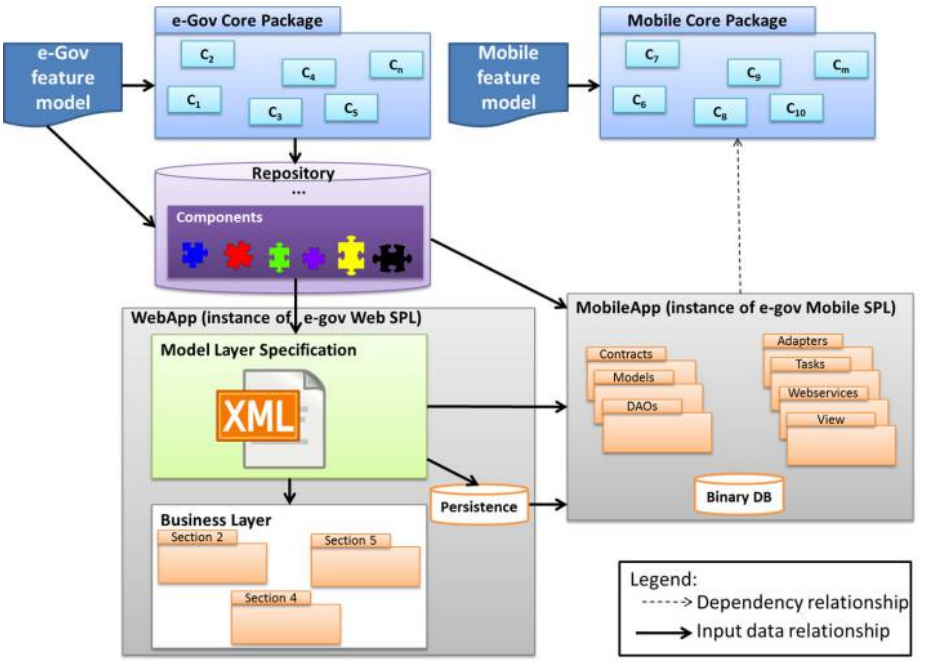


Figure 4.
Overview of the instantiation of the e-Gov Mobile SPL

name that entity will have in the mobile application (in the example, model = "Embassy").

Therefrom, as illustrated on Figure 4, many artifacts (Contracts, Models, DAOs, Adapters, Tasks, Webservices and Views) from the mobile application are generated based on the parameterization of functionalities from the Web application. If the application engineer also opts for the generation of the pre-loaded binary database for the mobile app, the Titan Architect Mobile extracts the data from the Web application (in PostgreSQL) converting it to make it available in the app (in SQLite). Finally, the classes from the Mobile Core Package, correspondents to features from mobile feature model, are copied into the mobile application's code, finalizing the instantiation process.

To present the results of the conducted evolution, a Web application (Embrapa, 2014a) (shown in Figure 6) was developed with the support of the e-Gov Web SPL and then corresponding mobile application was developed (Embrapa, 2014b) (illustrated in Figure 7) with the support of the e-Gov Mobile SPL, whose purpose is to provide a list of Brazilian embassies for overseas travelers. The development of both applications was carried out following the mentioned instantiation process and in accordance to the Application Engineering Process of the PLUS approach.

Thereby, first, an e-Gov Web SPL was instantiated. For this, it followed a Titan Framework step-by-step creation guide for Web applications (Carromeu, 2014), which involves the instantiation of the base code (the default code that all applications have, but can be modified by the application engineer) present in the framework's public repository and its configuration. Next, the list of Brazilian embassies was obtained in the Brazilian's Ministry of External Relations (commonly called Itamaraty) public website (Brasil, 2014), and imported into the Web application's database, in a table with columns corresponding to the quantity and types of the extracted data. It was created a new CRUD (create, read, update and delete) section in the Web application to enable the representation and manipulation of data from the table. Once the section was created, it was just necessary to enable its integration to the service bus through the parametrization of the corresponding component and available in the Titan Framework repository. Finally, it was activated in the Web instance the integration with the Google Plus API, allowing users to register and have access to the system using their Google accounts.

The second step was the creation of the mobile application, following the instantiation steps described in the beginning of this section for the e-Gov Mobile SPL. For this, the application engineer indicated functionalities from the Web application that

```
<?xml version="1.0" encoding="utf-8"?>
<mobile-source>
   <application name="Travel"
      package="com.carromeu.titan.sample.travel">

      <entity
         active="true"
         model="Embassy"
         section="embassy"
         xml="api-get.xml"
         table="embassy"
         assets="true"
      />

   </application>
</mobile-source>
```

Figure 5.
Specification of functionalities of the mobile application

Figure 6.
Web application of Brazilian embassies for overseas travelers

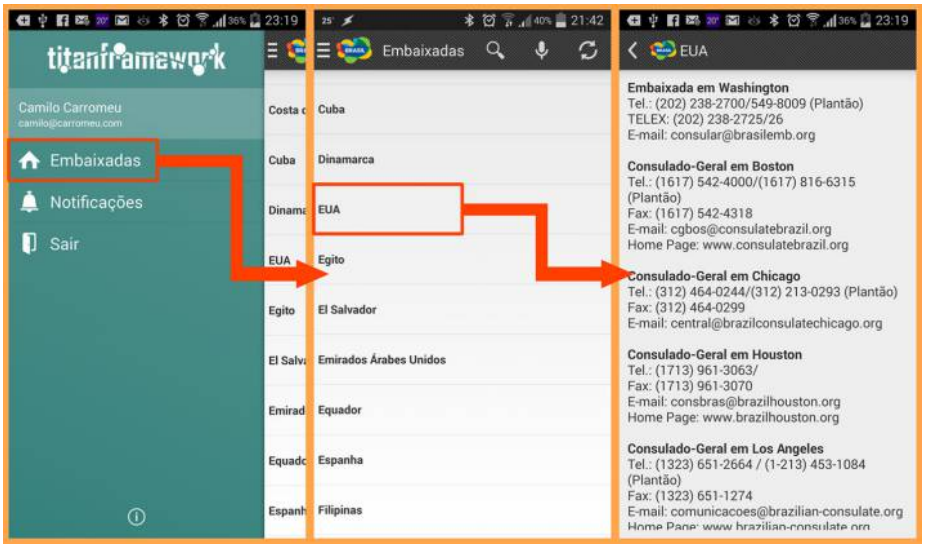


Figure 7.
Mobile application of Brazilian embassies for overseas travelers

should be present in the mobile application by means of a specification in XML (as illustrated in Figure 5). In the sequence, the Titan Architect Mobile was executed, generating the Contract, Model, DAO, Adapter, Task, Web services classes, XMLs and classes for the visualization layer as well, corresponding to the CRUD section of representation and manipulation for the list of embassies. The classes were generated in the packages and folders of the base code, including in it the functionality of synchronizing the embassies list in the mobile device. Finally, graphic alterations were made in the already functional mobile application (Figure 7) so that it became correspondent to the web application visual identity (Figure 6).

It is noted that a single application engineer developed both applications (Web and Mobile) and spent about 16 hours, being half of that time used in the creation of graphic elements. The optimization of development time was possible because of automatic code generation, which also helped to guarantee the final quality of the developed product.

6. Case studies

Several real-world solutions, cited below, were obtained based on the same instantiation process mentioned in Section 5 and are therefore members of the e-Gov Mobile SPL:

- *Livestock management platform*: This software platform is composed by a Web application (Titan instance) and a mobile application, whose code was generated from the instantiation of the e-Gov Mobile SPL. This solution allows beef cattle producers to manage the herd on their rural properties. The mobile application makes integral use of the services mentioned in Section 5.1, once it adopts a policy of asynchronous synchronization, where the user can interact with the mobile application while offline, creating tuples, identified making use of the disambiguation technique, that will be synced afterwards with the Web application using the service bus authenticated in application and client levels, provided in the Embrapa-Auth protocol.

- *Sanitary calendar*: The objective of this application is to notify the beef cattle producer about sanitary managements and vaccinations that the producer will have to apply in the herd. For this, the mobile application makes use of the Android notification system, named Google Cloud Messaging, which allows alerts to be launched to the mobile device. With that, from the record of business rules in the Web application, such as the correct age to apply specific vaccines in the animals and the registration of animal lots by the producer, the mobile application emits notices that allow the user to schedule coming managements and vaccinations that should be done.
- *Summary of bulls from the Geneplus program*: The Geneplus program (http://geneplus.cnpgc.embrapa.br/) is specialized service for animal genetic improvement available for the cattle breeder. One of the results of this program is a catalog that is annually updated with certified bulls (reproducers). This way, the developed mobile application aims to replace the catalog's distribution format that, until then, was printed. Then, this application's database was populated with about 70,000 animals, which compose the catalog. This mobile application remains synchronized with the corresponding Web application, providing the user with up-to-date information from the catalog, as well as other information like a search engine for bulls with specific characteristics.
- *Serviço de atendimento ao cidadão (SAC) Gado de Corte (Souza et al., 2012)*: In English, Customer Services Center for Beef Cattle, in English, is a mobile application with 1,700 questions and answers about the Brazilian chain of beef production.

Among the mobile applications developed from the instantiation of the e-Gov Mobile SPL, it is highlighted the mobile application SAC Gado de Corte that is available for free in the Google Play Store (Embrapa, 2012) since October of 2012, being available for any person with an smartphone or tablet with the Android operating system. It is noteworthy that until the moment, the application has more than 6,700 downloads.

The demand for the development of the mobile application SAC Gado de Corte has historical roots. The Brazilian Agricultural Research Corporation (Embrapa), since the 1970s, made use of medias for the knowledge transfer from its researchers to the rural producers community. In this period, doubts were sent by rural producers and answered by the Embrapa's researchers via letters. Over the years, this process kept evolving, following the technology, and today digital medias are used. Aiming to facilitate the communication with its public, the access and organization of this information, the SAC (Citizen's Service) was developed in many decentralized Embrapa units. In the Embrapa Beef Cattle, a Web application named SAC (Embrapa, 2013) was developed with support of the e-Gov Web SPL, as in the main screen illustrated on Figure 8. This Web application has a database with about 1,700 doubts, in the form of questions and their respective answers about the Brazilian livestock. In this context, aiming the high availability of this information to the producers, it led to the creation of a mobile application that enables the consultation of the database of the SAC Web application by smartphones and tablets.

Initially, the mobile application SAC Gado de Corte, previously installed in a mobile device [Figure 9(a),] synchronizes with the Web application SAC's database, storing the data

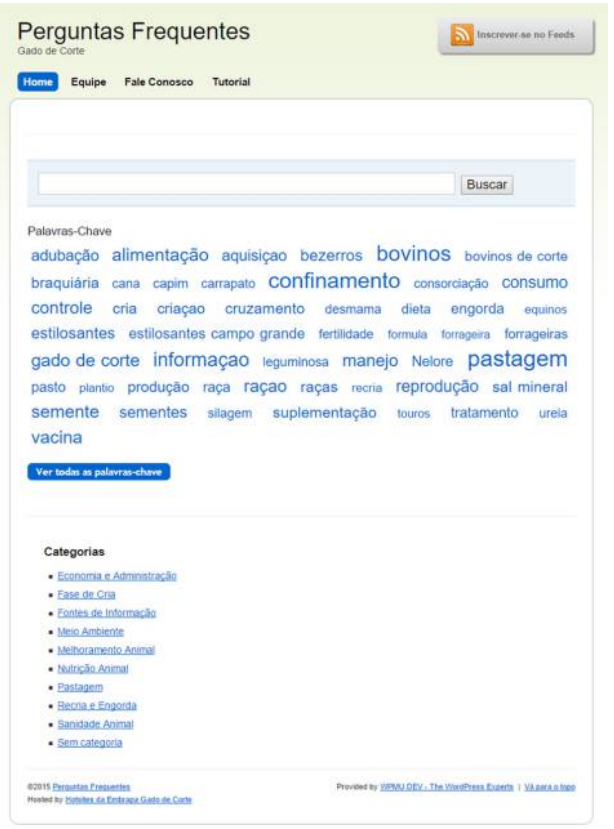


Figure 8.
SAC Web application

in the mobile device [Figure 9(b)]. The local database is updated every time the user restarts the mobile application (as long as there is a Internet connection available). After the synchronization, the user can search the whole local database. To do this, the user must interact with the mobile application accessing the questions of his/hers interest, which are organized by category, as illustrated on Figure 9(c).

When a category is selected ("Pastagem", for example), only questions of that category are shown. When a question is selected, a new window is opened, where the user can view the answer to that question, as illustrated on Figure 10(a and b) shows information about the development of the SAC Gado de Corte (laboratory, institution, license and authors).

Besides that, the user can filter the list of questions to find more easily what he/she is looking for. The mobile application SAC Gado de Corte has three ways to search a question:

(1) *Typed question*: In this type of search, the user is presented with a virtual keyboard, by which he/she can type one or more words related to the subject that he/she is searching, as shown on Figure 11(a).

(2) *Gestural search*: In this type of search, the user can draw characters on the screen related to the subject he/she aims to find, filtering the list of questions and answers, as illustrated on Figure 11(b).

Figure 9.
Starting screens of the mobile application SAC Gado de Corte

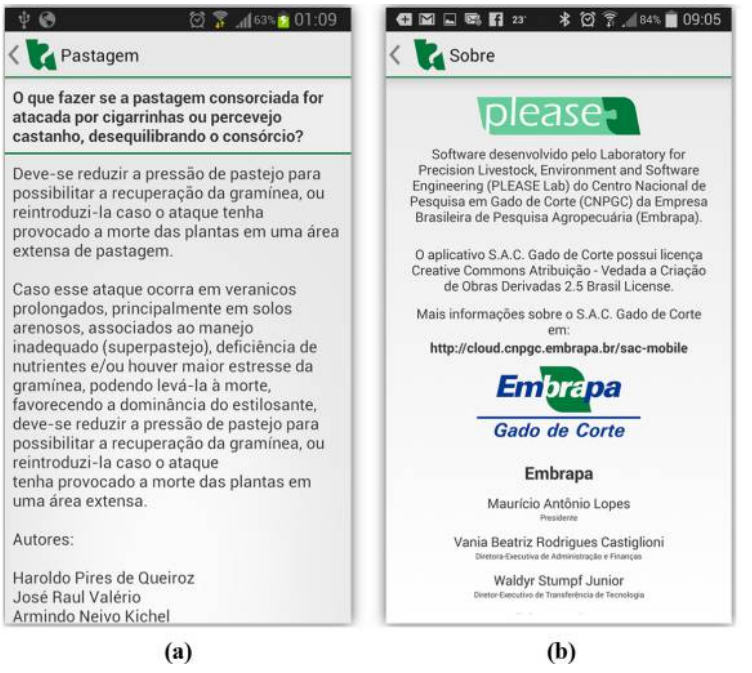


Figure 10.
Screen of the answer to a question related to the "Pastagem" category

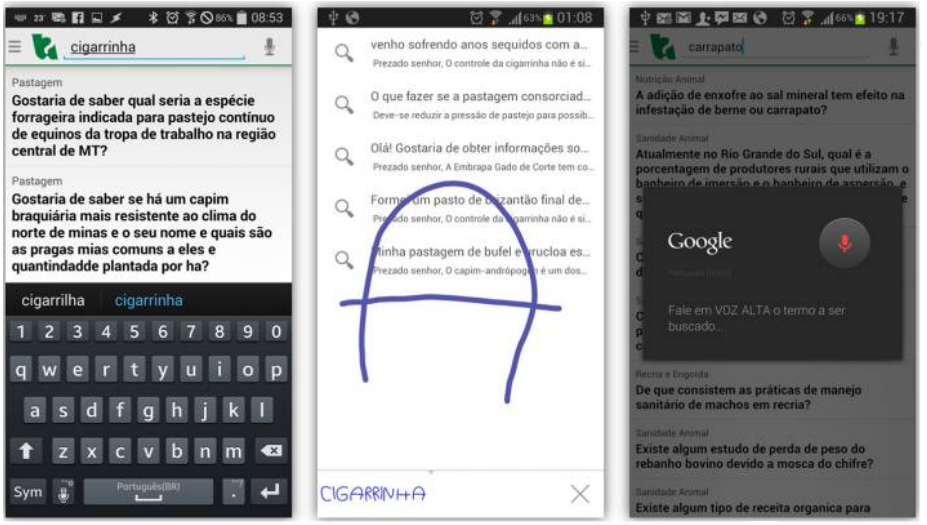


Figure 11.
Search types available in the mobile application SAC Gado de Corte

(3) *Voice search*: In this type of search, the mobile application allows the user to do a voice search, that is, the user speaks the subjects to be searched and the application recognizes and filters the list of questions and answers, as presented on Figure 11(c).

As shown on Figures 9-11, the instance of the e-Gov Mobile SPL has an interface with high usability focused on satisfactorily serving all of the Embrapa's target audience (researchers, students, rural producers, agriculture technicians, etc.).

Since its release, in the October of 2012, the mobile application SAC Gado de Corte has been used by diverse actors in the beef cattle production chain, as illustrated in Figure 12, once that its content is of interest of both agricultural technicians and rural producers. As it can be observed in the figure, there is an increase of 150 per cent on the number of users of the mobile application from 2014 to 2015, showing its visibility and utility for the agriculture and livestock on Brazil. It is noticed that besides Brazil, there is interest for the mobile application from users from others countries, as shown in Figure 13.

Based on the aforementioned data, it is possible to observe that the mobile application SAC Gado de Corte, a result from the instantiation of the e-Gov Mobile SPL, is equivalent in technical quality to other softwares for mobile devices that were developed completely in the manual way or generated by other tools.

7. Conclusion

This paper presented the changes made in the e-Gov SPL to obtain an e-Gov Mobile SPL. In particular, a service bus that contemplates the communication between Web applications and their corresponding mobile applications was implemented in the Titan Framework (some of the existing components in its repository were refactored to be able to interact with the incorporated services) and a code generator to create mobile

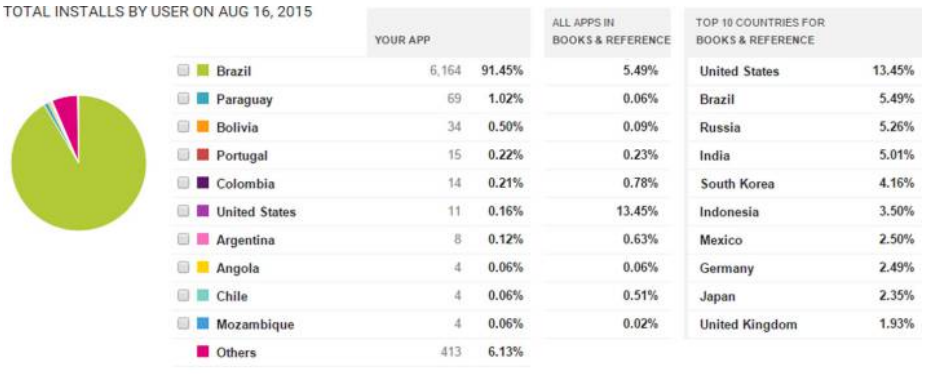
applications (Titan Architect Mobile) was implemented. The main motivation on conducting this work was the need to migrate several web systems, previously developed with the support of the e-Gov Web SPL, to a mobile platform, besides allowing the communication between the applications in both platforms (that is, Web and mobile).

From this work, it is possible to reduce time and effort during the development of Android applications and their integration with a web application that acts as a centralized database, and to ensure the adoption of best practices following Google's performance guidelines by mobile applications developed with Titan Architect Mobile. Some software products resulting from the e-Gov Mobile SPL were obtained up to the moment, highlighting the mobile application SAC Gado de Corte that had a significant increase in number of users, from different countries, in the past two years according to the data collected in the Google Play Developer Console.

As the main restrictions for this work, we can point out the small number of instantiations performed until the moment with the support of the e-Gov Mobile SPL, what prevented the conduction of an empirical study. However, the results obtained so far indicate considerable reduction in development time of mobile applications

Figure 12.
Number of unique users over time

Source: Obtained from Google (2015)



	YOUR APP		ALL APPS IN BOOKS & REFERENCE	TOP 10 COUNTRIES FOR BOOKS & REFERENCE	
Brazil	6,164	91.45%	5.49%	United States	13.45%
Paraguay	69	1.02%	0.06%	Brazil	5.49%
Bolivia	34	0.50%	0.09%	Russia	5.26%
Portugal	15	0.22%	0.23%	India	5.01%
Colombia	14	0.21%	0.78%	South Korea	4.16%
United States	11	0.16%	13.45%	Indonesia	3.50%
Argentina	8	0.12%	0.63%	Mexico	2.50%
Angola	4	0.06%	0.06%	Germany	2.49%
Chile	4	0.06%	0.51%	Japan	2.35%
Mozambique	4	0.06%	0.02%	United Kingdom	1.93%
Others	413	6.13%			

Figure 13.
Number of unique users by country

Source: Obtained from Google (2015)

corresponding to Web applications obtained through the e-Gov Web SPL and, consequently, cost reductions and increased productivity for application engineers. Moreover, the generated source code specifically works with the patterns adopted by Titan, that is, it must necessarily use the Embrapa-Auth, for example.

Main future works arising from this paper may include release of other services in the e-Gov SPL Mobile which will be identified with the usage of product line; addition of support to the code generation for iOS and Windows Phone in the Titan Architect Mobile; the development of an expressive number of mobile applications through the instantiation of the SPL treated in this paper, aiming to obtain statistically significant results and the conduction of controlled experiments that would allow to verify the effectiveness of the e-Gov SPL Mobile.

References

Actionbarsherlock (2012), available at: http://actionbarsherlock.com/ (accessed September 2015).

Android (2015), "Best practices", available at: http://developer.android.com/guide/practices/index.html (accessed September 2015).

Brasil (2014), "Ministry of foreign relations", available at: www.itamaraty.gov.br/ (accessed June 2014).

Buschmann, F., Meunier, R., Rohnert, H., Sommerland, P. and Stal, M. (1996), *Pattern-oriented Software Architecture – A System of Patterns*, Wiley & Sons, New Jersey.

Carromeu, C. (2014), "Titan framework cookbook", available at: http://cloud.cnpgc.embrapa.br/titan/documentacao/ (accessed September 2015).

Carromeu, C., Paiva, D.M.B. and Cagnin, M.I.C. (2015), "The evolution from a web SPL of the E-Gov domain to the mobile paradigm", *Proceedings of the 15th International Conference on Computational Science and Its Application, 2015*, Lecture Notes in Computer Science, London, Vol. 9155, pp. 217-231.

Carromeu, C., Paiva, D.M.B., Cagnin, M.I.C., Rubinsztjn, H.K.S., Breitman, K. and Turine, M.A.S. (2010), "Component-based architecture for e-Gov web systems development", *Proceedings of the 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems in Oxford, 2010*, IEEE, New York, pp. 379-385.

Cleaveland, J.C. (1988), "Building application generators", *IEEE Software*, Vol. 5 No. 4, pp. 25-33.

Clements, P. and Northrop, L. (2002), *Software Product Lines: Practices and Patterns*, Addison-Wesley, Boston, MA.

Coplien, J.O. (1998), "Software design patterns: common questions and answers", in Rising, L. (Ed.), *The Patterns Handbook: Techniques, Strategies, and Applications*, Cambridge University Press, New York, NY.

Embrapa (2012), "SAC Gado de Corte", available at: https://play.google.com/store/apps/details?id=br.embrapa.cnpgc.sac&hl=pt_BR (accessed September 2015).

Embrapa (2013), "SAC web application", available at: http://cloud.cnpgc.embrapa.br/sac/ (accessed September 2015).

Embrapa (2014a), "Embassy web application created from e-Gov Web SPL", available at: http://titan.cnpgc.embrapa.br/sample/travel/ (accessed September 2015).

Embrapa (2014b), "Embassy mobile application created from e-Gov Mobile SPL", available at: https://play.google.com/store/apps/details?id=com.carromeu.titan.sample.travel (accessed September 2015).

Falvo Junior, V., Duarte Filho, N.F., Oliveira Junior, E. and Barbosa, E.F. (2014), "A contribution to the adoption of software product lines in the development of mobile learning applications", *Proceedings of the IEEE Frontiers in Education Conference in Madrid, Spain, 2014*, IEEE, New York, pp. 1-8.

Foote, B. and Johnson, R.E. (1988), "Designing reusable classes", *Journal of Object Oriented Programming*, Vol. 1 No. 2, pp. 22-35.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995), *Design Patterns – Elements of Reusable Object-oriented Software*, Addison-Wesley, Boston, MA.

Gomma, H. (2005), *Designing Software Product Lines with UML*, Addison-Wesley, Boston, MA.

Google (2015), "Google play developer console", available at: https://play.google.com/apps/publish/signup/ (accessed September 2015).

ISO/IEC 9126:2001 (2001), "Software engineering – product quality – Part 1: quality model", June.

ISO/IEC 25010:2011 (2011), "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – system and software quality models", March.

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, CMU/SEI-90-TR-21, November.

Marinho, F.G., Andrade, R.M.C. and Werner, C.A. (2011), "Verification mechanism of feature models for mobile and context-aware software product lines", *Proceedings of the 5th Brazilian Symposium on Software Components, Architectures and Reuse in São Paulo, SP, Brazil, 2011*, IEEE, New York, pp. 1-10.

Marinho, F.G., Andrade, R.M.C., Werner, C., Viana, W., Maia, M.E.F., Rocha, L.S., Teixeira, E., Filho, J.B.F., Dantas, V.L.L., Lima, F. and Aguiar, S. (2013), "MobiLine: a nested software product line for the domain of mobile and context-aware applications", *Science of Computer Programming*, Vol. 78 No. 12, pp. 2381-2398.

Mizouni, R., Matarb, M.A., Mahmoudb, Z.A., Alzahmib, S. and Salahc, A. (2014), "A framework for context-aware self-adaptive mobile applications SPL", *Expert Systems with Applications*, Vol. 41 No. 16, pp. 7549-7564.

Nascimento, L.M., Almeida, E.S. and Meira, S.R.L. (2008), "A case study in software product lines - the case of the mobile game domain", *Proceedings of the 34th Eurocmicro Conference Software Engineering and Advanced Applications in Parma, Italy, 2008*, IEEE, New York, pp. 43-50.

Pascual, G.C., Herrejon, R.E.L., Pinto, M. and Fuentes, L. (2015), "Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications", *The Journal of Systems and Software*, Vol. 103, pp. 392-411.

Pohl, K., Bockle, G. and Linden, F.J.V. (2005), *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer-Verlag, London.

Quinton, C., Mosser, S., Parra, C. and Duchien, L. (2011), "Using multiple feature models to design applications for mobile phones", *Proceedings of the 15th International Software Product Line Conference*, *Munich, Germany, 2011*, ACM, Los Angeles, Vol. 2, pp. 1-8.

Sayyad, A.S., Menzies, T. and Ammar, H. (2013), "On the value of user preferences in search-based software engineering: a case study in software product lines", *Proceedings of the 35th International Conference on Software Engineering in San Francisco, CA*, IEEE, pp. 492-501.

SlidingMenu (2014), avaliable at: https://github.com/jfeinstein10/SlidingMenu (accessed September 2015).

Sommerville, I. (2010), *Software Engineering*, Addison-Wesley, Boston, MA.

Souza, D.C.G., Righes, B., Rodrigues Filho, J.R., Queiroz, H.P. and Carromeu, C. (2012), "Mobile service for citizen: SAC mobile", *Proceedings of the 8th Scientific Meeting of the Embrapa Beef Cattle, Campo Grande, MS, Brazil*, pp. 120-121.

Weiss, D.M. and Lai, C.T.R. (1999), *Software Product-Line Engineering: A Family-based Software Development Process*, Addison-Wesley Professional, Boston, MA.

About the authors

Camilo Carromeu received the BS degree in Computer Science from the Federal University of Mato Grosso do Sul in 2003, the MS degree in Computer Science from the Federal University of Mato Grosso do Sul in 2007. He is currently an Information Technology Analyst at Brazilian Agricultural Research Corporation – Embrapa Beef Cattle. His research interests include Software Engineering, Precision Agriculture and Livestock, Web Engineering, Geographic Information System and Reuse Techniques (Framework, Software Product Line and Application Generator).

Débora Maria Barroso Paiva received the BS degree in Computer Science from the Federal University of Ouro Preto in 1998, the MS degree in Computer Science and Computational Mathematics from the University of São Paulo in 2001 and the PhD degree in 2008 in Computer Science and Computational Mathematics from the University of São Paulo. She is currently an Associate Professor at the Federal University of Mato Grosso do Sul. Her research interests include Software Engineering and Hypermedia.

Maria Istela Cagnin received de BS degree in Data Processing Technology from Fundação Paulista de Tecnologia e Educação in 1995, the M.S. degree in Computer Science from the Federal University of São Carlos in 1999 and the PhD degree in Computer Science and Computational Mathematics from the University of São Paulo in 2005. She is currently an Associate Professor at the Federal University of Mato Grosso do Sul. Her research interests include Software Engineering Processes, Reengineering, Reuse Techniques (Frameworks, Software Patterns and Software Product Line) and Business Model Reuse. Maria Istela Cagnin is the corresponding author and can be contacted at: istela@gmail.com