The Journal of Systems and Software 123 (2017) 1–32

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A product-line model-driven engineering approach for generating feature-based mobile applications

Muhammad Usman*, Muhammad Zohaib Iqbal, Muhammad Uzair Khan

Software Quality Engineering and Testing (QUEST) Laboratory, National University of Computer and Emerging Sciences, Islamabad, Pakistan

ARTICLE INFO

Article history:
Received 30 June 2016
Revised 31 August 2016
Accepted 28 September 2016
Available online 29 September 2016

Keywords:
Mobile applications
Software product-line engineering
Feature model

ABSTRACT

A significant challenge faced by the mobile application industry is developing and maintaining multiple native variants of mobile applications to support different mobile operating systems, devices and varying application functional requirements. The current industrial practice is to develop and maintain these variants separately. Any potential change has to be applied across variants manually, which is neither efficient nor scalable. We consider the problem of supporting multiple platforms as a 'software product-line engineering' problem. The paper proposes a novel application of product-line model-driven engineering to mobile application development and addresses the key challenges of feature-based native mobile application variants for multiple platforms. Specifically, we deal with three types of variations in mobile applications: variation due to operation systems and their versions, software and hardware capabilities of mobile devices, and functionalities offered by the mobile application. We develop a tool MOPPET that automates the proposed approach. Finally, the results of applying the approach on two industrial case studies show that the proposed approach is applicable to industrial mobile applications and have potential to significantly reduce the development effort and time.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

Mobile application development has recently emerged as one of the most focused areas in the software industry (Dehlinger and Dixon, October, 2011). Exponential growth of mobile users, extensive use of mobile devices, and the variety of mobile platforms has resulted in significant increase of mobile application development industry. According to recent statistics, there are around 4 million registered mobile applications available for various platforms (Statista 2015) with approximately 40 thousand mobile applications being added per month (Android Apps Submitted Per Month 2015) making the mobile application industry a multi-billion dollar industry (Mobile Application Revenue Generation 2013).

With the increase in the variety of mobile operating systems and device features, a typical challenge faced by mobile application industry is the requirement to support different mobile platforms. The term mobile platform refers to both the software platform (i.e., operating system and software features such as contact and message) and the hardware platform (i.e., the mobile device hardware such as bluetooth and camera). As an example, consider an application being developed may need to support various Android operating systems' versions (Google 2015) (such as, *Ice-cream Sandwich, Kitkat, Lollipop, Marshmallow*), their forks (Android fork such as *Cyanogen*), and various iOS versions (Apple 2013) (such as, *iOS 7, iOS 8, iOS X*). The same application may also need to support the mobile device hardware variations, i.e., various mobile devices support specific hardware features that are not available in other devices. Some mobile devices provide support for LTE/4 G Networks, Bluetooth, GPS, External storage, or Accelerometer and other devices do not support these.

Another requirement for mobile applications is to support multiple functional requirement variations, i.e., different functionality for different clients. For example in a banking application, some clients may require support of scheduled wireless backups, while other clients only require manually triggered wired backup support. Similarly, some clients are interested in payment via traditional credit card option only whereas others may want support for more novel options, such as, bitcoins or Near Field Communications payment systems, such as Apply pay (Apply Pay 2016) and Android pay (Android Pay 2016) in an e-store application.

A common mobile development industry practice to address the above mentioned problems is to develop separate native variants of the same application for each mobile platform (Dehlinger and Dixon, October, 2011; Joorabchi et al., October 2013; Top 100 Apps Availability for iOS 2016). This is a very resource heavy task and the complexity of maintaining multiple variants increases exponentially as a large number of application variants need to

* Corresponding author:.
E-mail addresses: m.usman@questlab.pk (M. Usman), zohaib.iqbal@nu.edu.pk (M.Z. Iqbal), uzair.khan@nu.edu.pk (M.U. Khan).

http://dx.doi.org/10.1016/j.jss.2016.09.049
0164-1212/© 2016 Elsevier Inc. All rights reserved.

be developed and maintained over time (Joorabchi et al., October 2013). Any potential change in the application requirements has to be applied to all the different variants of the mobile application manually. Similarly, any new functionality needs to be added in all the variants separately. The graphical user interface (GUI) for these applications is developed specifically for particular mobile platforms. The GUI is typically developed using drag-and-drop tools available for various operating systems and is tailored specifically for optimal performance on each set of devices (Google 2016; Microsoft, 2013; Apple 2016) that are then used to generate code corresponding to the GUI. Similarly, user-interface modeling languages, such as, IFML (OMG 2016), UML*i* (Da Silva and Paton, 2000), or model-based user-interface modeling techniques (Cimino and Marcelloni, 2012; Botturi et al., 2013; Sabraoui et al., 2012) can be used for developing GUI. The development of business logic of native variants requires redundant effort and the approach of developing these separately is neither scalable nor feasible. The problem of maintaining multiple variants has also been highlighted by a number of software engineers as one of the key challenges in mobile application development (Dehlinger and Dixon, October, 2011; Joorabchi et al., October 2013; Wasserman, 2010).

An alternative to developing multiple native variants of a mobile application is to either develop a web application or a cross-platform hybrid application by using web-scripting languages (Raj and Tolety, 2012). Web applications execute in a web browser, whereas hybrid applications execute inside native containers of mobile devices (Raj and Tolety, 2012). Web and hybrid mobile applications are generally considered low in performance and cannot access mobile device hardware directly (Charland and Leroux, 2011). Web applications also require internet connectivity that cannot be ensured at all times. Native applications, on the other hand, are considered to be more stable and secure, better in terms of performance, allow direct access to device hardware and also have better look and feel. For example, Facebook application was first launched as a hybrid application but due to the performance issues and hardware non-accessibility, later on it was developed as a native application for multiple platforms (Facebook Hybrid App to Native App 2015). Some cross-platforms tools exist for development of web and hybrid mobile application (Ohrt and Turau, 2012). The mobile applications developed using native application development tools (Google, 2016; Microsoft, 2013, Apple 2016) have superior look and feel as compared to the applications developed from the cross-platform development tools. Similarly, the debugging support offered by the cross-platform tools is inferior to the support provided by the native application development tools (Ohrt and Turau, 2012; Dalmasso et al., 2013; Heitkötter et al., 2012). Therefore, developing native applications is often the preferred choice if not the only choice.

Software product-line engineering (SPLE) is a well-accepted approach of developing a set of products that share a common set of features (Pohl et al., 2005). The concept of SPLE has been adopted from the broader product-line engineering that has been adopted and applied in different domains to handle large number of product variations, for example, in the automobile industry (Thiel and Hein, 2002) and embedded systems (Polzer et al., 2009). SPLE uses feature models that consist of a set of features and their variations that are required by the family of products being developed (Lee et al., 2002). The various products in the family (also referred to as product variants) in a product-line are derived using the feature model (Webber and Gomaa, 2004). The problems of supporting mobile application variants that are highlighted above can be positioned as an SPLE problem and the existing SPLE concepts can be applied in this domain.

A widely accepted methodology for developing software systems that has successfully been applied in other domains is Model-Driven Software Engineering (MDSE) (Gomaa, 2008; Larman, 2004; Iqbal et al., 2015). In MDSE, models are considered as the key software development artifact (Brambilla et al., 2012) and Unified Modeling Language (UML) (OMG 2013a) is commonly used for developing software systems. MDSE has a high potential for use in developing mobile applications because it allows platform independent modeling, which can later on be transformed to multiple mobile platforms.

In our work, we address the challenges faced by the mobile application industry (highlighted earlier) by proposing a 'product-line model-driven engineering approach' to support automated generation of mobile application variants of multiple platforms. We refer to these variants as feature-based variants and our approach supports three variations required by mobile applications industry: (i) variations in application due to mobile operating systems and their versions, (ii) variations due to software and hardware capabilities of the mobile devices, and (iii) variations based on the functionalities offered by the mobile application.

The problem of supporting multiple mobile application variants (hardware, software, and functional) is well-acknowledged in literature and is a recurring problem in industry. This is also true for our industrial partner, Invotyx (Invotyx 2014), which is developing native mobile applications for various mobile platforms with different features and the developers are currently maintaining a number of variants for each application, as per the prevalent industrial practice. Invotyx is interested in an efficient approach for developing and maintaining the various variants of the applications being developed at the company. Our proposed approach provides an automated solution and is applied on two case studies provided by our industrial partner, *Scramble* and *Instalapse*.

As part of the solution, for SPLE we provide a generic mobile application product-line feature model (FM_G) that captures the mobile domain specific concepts (for example, Android v5.1, iOS X, bluetooth, WIFI, contact, and message). The FM_G can be used by the application designer to develop an application specific feature model (FM_A). The FM_A contains the operating system-related features, software-related features, hardware-related features, and application-specific functional requirement-related features and combine these as a feature model specific for the application product-line under development. The application specific feature model (FM_A) can then be used to generate a mobile application product-line modeling profile specific for the application product-line. The modeling profile allows the application designer to model mobile domain specific concepts during mobile application modeling. Considering mobile applications typically have short time to market and require quick delivery and deployment (Joorabchi et al., October 2013), we select a minimal but sufficient subset of UML that includes UML use-case diagram for requirements gathering, class diagram for structural modeling, and state machine diagram for behavioral modeling of the mobile application. To support the proposed approach, we develop an application generation tool MOPPET to automate our product-line model-driven engineering approach for mobile applications. The proposed approach is applied on two industrial case studies developed by our industrial partner.

The rest of the paper is organized as follows: Section 2 highlights the industrial context and describes the case studies. Section 3 overviews the related work. Section 4 presents the details of the proposed approach briefly. Section 5 provides details about the proposed product-line engineering approach while Section 6 discusses the proposed model-driven engineering approach. Section 7 describes the feature-based mobile application variants generation while Section 8 presents the developed MOPPET tool to implements the proposed approach. Section 9 validates the proposed approach through industrial case studies and also highlights the limitations, threats to validity, and open questions of the proposed approach. Finally, Section 10 concludes the paper.

Download English Version:

https://daneshyari.com/en/article/4956603

Download Persian Version:

https://daneshyari.com/article/4956603

Daneshyari.com