8-2015

In Need of a Domain-Specific Language Modeling Notation for Smartphone Applications with Portable Capability

Hamza Ghandorh
Western University, hghandor@uwo.ca

Luiz Fernando Capretz Dr.
Western University, lcapretz@uwo.ca

Ali Bou Nassif Dr.
Western University, abounas@uwo.ca

Citation of this paper:

1. Ghandorh H., Capretz L.F. and Nassif A.B. In Need of a Domain-Specific Language Modeling Notation for Smartphone Applications with Portable Capability, 12th International Conference on Mobile Web and Intelligent Information Systems, Rome, Italy, pp. 218-227, Lecture Notes in Computer Science (LNCS 9228), Springer, DOI: 10.1007/978-3-319-23144-0_20, August 2015.

A Domain-specific Language Notation for Smartphone Applications with Portable Capability

Hamza Ghandorh, Ali Bou Nassif, and Luiz Fernando Capretz

Department of Electrical and Computer Engineering
The University of Western Ontario
London, Ontario, Canada
{hghandor,abounas,lcapretz}@uwo.ca

Abstract. A rapid growing of smartphone market and its increasing revenue have developers to target multiple platforms. Each leading software company, e.g. Apple or Microsoft, develops its smartphone applications or apps complying with its own specifications. The specification of each platform makes a platform-dedicated application incompatible with other platforms due to the diversity of operating system, programming language, and design patterns. As a consequence, development of dedicated applications for multiple platforms is tedious task. Conventional development methodologies are applied to smartphone apps, but less performance and requirements appear which reduce their quality. Such phenomena occurred due to two perspectives: unique hardware and software requirements. Several previous works considered automatically generating executable code based on abstract models that would alleviate platforms fragmentation. It is possible that defining smartphone applications considering portability requirements using a customize notation would contribute to smartphone app quality. This paper proposes a domain-specific language notation to design portable smartphone applications using appropriate abstractions.

Keywords: Smartphone applications, DSL, Modeling, Portability

1 Introduction

Smartphone apps become a vital part of our lives due to its contributions. The more smartphone apps has become innovative, the more users see them as a desirable asset. This demand triggered the intense competition among the major smartphone companies, namely Apple, Google, Microsoft and Blackberry to provide more innovative apps. This competition not only had enabled a rapid growth in mobile market and the emergence of increasingly better features, but also was responsible for software development complexity and mobile platform fragmentation. Each leading software company, such as Google, Apple or Microsoft, produces its smartphone apps complying with its own designing and implementation specifications with their specific tools [1]. The specification of each platform makes developed applications for a given platform incompatible with other platforms due to the diversity of operating system, programming language, and design patterns. This lack of compatibility have smartphone developers to rewrite their application for each one of the target platforms increasing the effort and the time to market of that application. Hence, dedicated applications development for each platform is a non-trivial task for software engineers when considering labour and maintenance costs [2].

Similarly as other type of software, smartphone apps may need to migrate to a variety of platforms due to the growing diversity of computing environments over its lifetime [2]. In the end of 2014, Android was the dominate platform for smartphone development where it hold 71% of developers [1]. After only few months, Android was targeted by only 40% of professional developers, where other platforms gained more priority, such as iOS for 37%, Windows Phone and the mobile browser have just 8% and 7% of developers due to users requirements, respectively [1]. Software developers agree that application portability is a desirable attribute for their software projects due to durable cost-effectiveness and for a maximum of end-users [2][3]. The primary goal of portability is to facilitate the activity of *porting* an application from an environment in which it currently operates to a new or target environment prior to allow reuse of the complete existing application in the new environment [2]. Concerns in application portability include maintaining quality as well as saving time and money, and in leveraging an existing effort in the deployment of software design

in new ways [2].

It is possible that defining smartphone applications considering portability requirements by using appropriate notations and automatically generating executable code will alleviate mobile platform fragmentation. One approach was used in the literature referred to as Domain-specific language (DSL)[4][5]. DSL aims to express solutions at the level of abstraction of the problem domain based on thorough understanding of the application domain. It specifies a software solution in a language that directly uses concepts and rules from the specific problem domain, and it generates final products in a chosen programming language or other form from these high-level specifications [5].

2 Problem and Motivation

Smartphone developers have used several approaches to target multiple platforms. One approach is to develop apps using cross-platform tools such as PhoneGap[1] or Xamarin[2]; other approach is to develop smartphone apps based on web-technologies (e.g. HTML5 or JavaScript) or cloud technology; or developing apps by using abstraction layers that maps write-once code to be generated to many platforms [19]. Although these approaches try to cope with the problem of smartphone platform fragmentation and provide smartphone apps with portable capabilities, each approach has its own caveats in term of high cost, delivering less functional requirements or inefficient usage of smartphone hardware capabilities [19].

Although smartphone design seems independent of its implementation and it should be perfectly reused by definition, the chosen design method will have a major impact on smartphone apps portability in term of architecture design, GUI and controlling other direct and indirect interfaces [2]. Conventional software development methodologies are proven to be effective for desktop software products, and these methodologies are applied to smartphone apps. However, less performance and requirements appear on smartphone apps products which reduce their quality [20]. Such phenomena occurred due to two perspectives: unique hardware and application requirements. Smartphone devices could be expanded to new and several type of hardware, and they enable rich user interface input which increases apps operability more then desktop devices [21][22]. Smartphone devices differentiate in screen size, input/output facilities, and their graphical user interface (GUI) which usually needs to be significantly adjusted [23]. Several special considerations need to be made for smartphone apps development. For example, smartphone apps is built in very short time with low prices, they operate on constantly event-driven, their life cycle is very limited, and their quality depends on its GUI responsiveness and its efficiently to save battery life [14]. All previous requirements should be considered in the design phase and they impose own challenges against porting activity between different smartphone platforms.

Another issue that negatively impact the quality of smartphone apps is that smartphone apps are designed with superficial or ad-hoc design approaches. Smartphone developers do not follow a systematic standard in designing their apps where they create mockups with simple and basic graphics and produce dummy version of their apps that include UI screen and element interactions [24].

3 Related Work

Several research aimed to examine software quality in smartphone platforms in term of quality testing models such as [6], or non-functional quality attributes such as smartphone usability [7][8], smartphone reusability [9], or smartphone reliability [10].

Several studies have used several notations such as Domain-specific language (DSL)[4], extension of Unified Modeling Language (UML[3]), or other modeling notations or tools to model smartphone apps to target multiple platforms. Some studies used Domain-specific language (DSL) such as X_{MOB}[11], and MobDSL[12]. These studies used DSL to generate executable code to

[1] PhoneGap site `http://phonegap.com/`
[2] Xamarin site `http://xamarin.com/`
[3] UML site `http://www.uml.org/`

different platforms. They always narrow down to a specific platform and did not concern about quality design with portable capabilities. Other studies used extension of UML notation, such as UML2 profile[13], Android-related UML model [14], and Windons 7-related UML model[15]. Another work used model-driven engineering based on mix of UML and DSL[16] or Interaction Flow Modeling Language (IFML)[4] extension to build smartphone front-end [17]. Also, another work is referred to as ApplDE [18] where they use Software Product Line (SPL) approach. ApplDE use model-driven engineering and SPL to bridge the gap between business variability and device variability. These studies used UML and other modeling notations that concern about static structure or class diagrams of a smartphone app, but they do not consider smartphone apps requirements such as user interface (UI) design specifications.

All previous works pursued targeting multiple platforms after the implementation phase of smartphone apps and omitting design models. Our work paid more attention to providing design models that comply with smartphone apps specifications and portability requirements.

4 Proposed Solution

We propose a notation to define smartphone applications using appropriate abstractions. We are planning to maintain our version of domain-specific language (DSL) considering three main portability requirements: 1) to identify architecture design, 2) to identify software dependencies, and 3) to identify supported features in a given platform. The notation should allow apps designers or developers to draw architecture design where it entail direct interfaces, such as I/O storage and devices interfaces, and indirect interfaces, such as user interface. The notation should allow developers to draw software dependencies, such as external APIs and common libraries. Moreover, the notation should list all common basic features supported by each target platform such as user interface (UI) features, and user experience (UX) gestures.

In order to achieve our goal, we need to maintain two components in our notation: DSL ontology and DSL meta model or domain model. The DSL ontology will be used to represent a set of concepts and relationships of smartphone domain. The ontology could be reused for further extension of our notation, where it will save software developers' time and efforts. The DSL domain model defines the concepts of a language and their relationships in a domain complying with the portability requirements. The domain model will be build based on the DSL ontology. In addition, smartphone app repositories will be build to include several architecture design and components, external APIs and common libraries, user interface (UI) capabilities, and user experience (UX) gestures.

Our solution targets two stakeholders in smartphone apps development: developers, and team managers. Apps developers will be able to visually model their apps, to update and reuse them, and to document rational behind his/her design. Team managers will be able to own a customize modeling language to maintain future projects on several platforms with in-advance knowledge of efforts and needed infrastructure. The notation should be familiar to apps developers with simple and informative constructs. The notation should allow for iterative addition and smooth evolution.

5 Current Status

Our notation is in-progress work and we are working on an initial prototype. After a prototype is completed, initial surveys will be submit to specialized developers in smartphone apps to examine and provide their feedback for the notation enhancement.

Acknowledgments The research for this paper was financially supported by the Ministry of Education of Saudi Arabia and King Abdullah Foreign Scholarship Program, and College of Computer Science and Engineering at Taibah University[5].

[4] IFML site `http://www.ifml.org/`
[5] Taibah University site `https://www.taibahu.edu.sa/Pages/AR/Home.aspx`

References

1. Economics, D.: Developer economics third quarter 2014 and first quarter 2015: State of the developer nation. https://www.developereconomics.com/reports/ (2014-2015)
2. Mooney, J.D.: Developing portable software. In Reis, R., ed.: Information Technology. Volume 157 of IFIP International Federation for Information Processing. Springer US (2004) 55–84
3. Johansson, A., Svensson, J.: Techniques for software portability in mobile development. Master's thesis (2009)
4. Fowler, M.: Domain Specific Languages. 1st edn. Addison-Wesley Professional (2010)
5. Kelly, S., Tolvanen, J.P.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley (2008)
6. Liu, Z., Hu, Y., Cai, L.: Software quality testing model for mobile application. In Awan, I., Younas, M., Franch, X., Quer, C., eds.: Mobile Web Information Systems. Volume 8640 of Lecture Notes in Computer Science. Springer International Publishing (2014) 192–204
7. Mesfin, G., Ghinea, G., Midekso, D., Grnli, T.M.: Evaluating usability of cross-platform smartphone applications. In Awan, I., Younas, M., Franch, X., Quer, C., eds.: Mobile Web Information Systems. Volume 8640 of Lecture Notes in Computer Science. Springer International Publishing (2014) 248–260
8. Nayebi, F., Desharnais, J.M., Abran, A.: An expert-based framework for evaluating ios application usability. In: Joint Conference of the 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement. (Oct 2013) 147–155
9. Mojica, I.J., Adams, B., Nagappan, M., Dienst, S., Berger, T., Hassan, A.E.: A large-scale empirical study on software reuse in mobile apps. IEEE Software **31**(2) (Mar 2014) 78–86
10. Meskini, S., Nassif, A.B., Capretz, L.F.: Reliability models applied to mobile applications. In: Proceedings of the 7th International Conference on Software Security and Reliability-Companion. (June 2013) 155–162
11. Goaer, O.L., Waltham, S.: Yet another dsl for cross-platforms mobile development. In: Proceedings of the First Workshop on the Globalization of Domain Specific Languages, New York, NY, USA, ACM (2013) 28–33
12. Kramer, D., Clark, T., Oussena, S.: Mobdsl: A domain specific language for multiple mobile platform deployment. In: 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications. (Nov 2010) 1–7
13. Botturi, G., Ebeid, E., Fummi, F., Quaglia, D.: Model-driven design for the development of multi-platform smartphone applications. In: 2013 Forum on Specification Design Languages. (Sept 2013) 1–8
14. Kraemer, F.A.: Engineering android applications based on uml activities. In: Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems, Berlin, Heidelberg, Springer-Verlag (2011) 183–197
15. Min, B.K., Ko, M., Seo, Y., Kuk, S., Kim, H.S.: A uml metamodel for smart device application modeling based on windows phone 7 platform. In: 2011 IEEE Region 10 Conference TENCON. (Nov 2011) 201–205
16. Ribeiro, A., da Silva, A.R.: Xis-mobile: A dsl for mobile applications. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, New York, NY, USA, ACM (2014) 1316–1323
17. Brambilla, M., Mauri, A., Umuhoza, E.: Extending the interaction flow modeling language {IFML} for model driven development of mobile applications front end. In Awan, I., Younas, M., Franch, X., Quer, C., eds.: Mobile Web Information Systems. Volume 8640. Springer International Publishing (2014) 176–191
18. Quinton, C., Mosser, S., Parra, C., Duchien, L.: Using multiple feature models to design applications for mobile phones. In: Proceedings of the 15th International Software Product Line Conference. Volume 2., New York, NY, USA, ACM (2011) 23:1–23:8
19. Wasserman, A.I.: Software engineering issues for mobile application development. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, New York, NY, USA, ACM (2010) 397–400
20. Rogness, N., Case, S.: An assessment of design and implementation trade-offs and their impact on mobile applications. In: Midwest Instruction and Computing Symposium. (2003)
21. Lee, J.S., Chae, H.: Domain-specific language approach to modelling ui architecture of mobile telephony systems. IEE Proceedings Software **153**(6) (Dec 2006) 231–240
22. Matsui, K., Matsuura, S.: {MDD} for smartphone application with smartphone feature specific model and {GUI} builder. In: The 9th International Conference on Software Engineering Advances. (Oct 2014) 64–68
23. Nystrom, L.: Computer Scientist Seeks to Improve Portability of Mobile Device Applications. http://goo.gl/4jAdMG (August 2012) Online; accessed 02-July-2014.
24. Reaza, A.: The 7 phases for good mobile ui design. http://goo.gl/JYWdHw (2013)