Identification of reusability artifacts in mobile application development process

Zlatko Stapić
University of Zagreb, Faculty of Organization and Informatics Varazdin
Laboratory for Applied Software Engineering
Pavlinska 2, HR 42000 Varazdin, Croatia
zlatko.stapic@foi.unizg.hr

***Abstract**. Development of mobile applications is a challenging task which differs from traditional software development, and needs to address the business and development specific challenges. Our research focuses on development specific challenges. The main goal of this research is to identify reusability artifacts in mobile application development process targeting two or more mobile platforms, and to create the basis for more efficient and interoperable cross-platform mobile development process. After identifying all artifacts in the development process for one target platform, we have performed the reusability analysis for the second platform. We found that 66% of artifacts are completely or partially reusable.*

Keywords. software development, mobile application development, development methodologies, reusability

1 Introduction

Traditional and mobile development differ in several important aspects which makes mobile development a challenging task. There are different classifications, but Hosbond (2005) focuses on the two main sets of challenges that should be addressed in mobile systems development. These are business related and development specific challenges. In this research we will focus on development specific challenges and will give special attention to the usage of methodologies which according to some authors, like Rahimian and Ramsin (2008), Spataru (2010) or La and Kim (2009), should be firstly addressed.

Classic or agile software development methodologies should be adapted for the development of mobile applications as the existing ones do not cover the specific mobile targeted requirements (La & Kim, 2009). There are attempts from several authors to create new methodologies in order to cover the gaps in the domain of mobile applications.

Along with the methodology related problems, the fragmentation problem forces the developers of mobile applications to focus on only specific platforms and versions (Shah et al., 2019), but as the development of mobile applications primarily aims a wide range of users, such approach is not the preferable option and the development teams reach for different solutions of the problem proposed by professional and scientific community. First, worth to mention is the approach that enables the development teams to use a mediatory language or just mediatory transform engine to code for several target platforms. Some of the most influential projects are React Native (Facebook Inc., 2020) and Flutter (Google Inc., 2020). Although, these attempts have some advantages they also have significant drawbacks, like their dependability on the efforts invested in the transform engine, specific APIs and specific domain, the lack of control over generated source code and similar. Another possible solution to the problem could be the introduction of adapter applications (adapters) as native applications for every target platform. According to Agarwal et al. (2009) this is one of the two main techniques for handling fragmentation. As standardization of APIs in mobile world is still not possible, the usage of programming techniques whereby the interface calls are wrapped, i.e. abstracted, in distinct modules which are then ported across the platforms, is left as the other solution. The representatives of this approach are MobiVine (Agarwal et al., 2009), Adobe PhoneGap (PhoneGap, 2020) or Adobe AIR® (Adobe Inc., 2020). Almost all of the drawbacks stated for existing solutions that introduce a transform engine are also present in this solution. Finally, the third approach is to use web technologies and to develop cross-platform web applications, but this approach is out of our scope as it differs in many aspects (which also have their own drawbacks) from the basic assumptions taken in this research.

Therefore, in our research we are focused on finding a solution that would enhance methodological interoperability among teams working on the same application but in different (and native) development environments. In the context of more comprehensive research, we aim to identify what artifacts (required inputs and outputs of methodologically and methodically defined development steps) emerge

during mobile application development, whether and to what extent there are similarities between these artifacts so they could be reusable during the development process for two or more mobile platforms, and to create the basis for more efficient and interoperable process of multi-platform mobile applications development.

The paper is structured as follows. In section two we bring the methodological context and artifacts analysis setup while in third section we present the identified artifacts and their types. Section four brings the reusability analysis results while the last section concludes the topic and looks forward to the future research.

2 Analysis setup

In our previous research (Stapic et al., 2016), we performed a literature review in order to identify newly created methodologies targeting the development of mobile applications and we have identified 14 different methodologies. However, only one of these methodologies was reported to be used in practice – Mobile-D.

Mobile-D process, as described in (P Abrahamsson et al., 2004; Supan et al., 2013; VTT Technical Research Centre of Finland, 2006), includes five phases that are executed in partially incremental order (see Fig. 1). The aim of the first phase, called Explore, is to prepare the foundation for future development. The Initialize phase should describe and prepare all components of the application as well as to predict possible critical issues of the project.

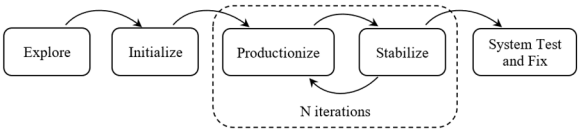

Figure 1. Mobile-D process *(VTT Technical Research Centre of Finland, 2006)*

The Productionize and Stabilize phases are executed iteratively in order to develop all other features of the mobile product. Iterations start with planning day in Productionize phase. The first activity is post-iteration workshop which aims to enhance the development process to better fit the needs of the current software development team. Requirements analysis, iteration planning and acceptance test generation tasks follow and are executed during the planning day. Working day is based on implementation through test driven development, pair programming, continuous integration and refactoring. This day ends with the task of informing the customer on new functionality. Finally, the release day includes the activities of integration and testing. The Stabilize phase has the goal to finalize the implementation along with integrating subsystems if necessary. As this phase can contain additional programming and development, the activities are very similar to the activities in the Productionize phase. The only additional activity concerns documentation wrap-up. Iterations should result in a working piece of functionality at user level. Finally, System Test and Fix phase aims to detect if the produced system correctly implements the customer defined functionality. It also provides the project team feedback on the systems functionality and the defect information for the last fixing iteration of the Mobile-D process. This last iteration is not obligatory, but when fixing is needed it consists of the same activities as other implementation iterations already explained (P Abrahamsson et al., 2004; Supan et al., 2013; VTT Technical Research Centre of Finland, 2006).

Mobile-D strongly suggests the usage of Test Driven Development (TDD) which is connected to all Mobile-D phases . The basics and the state of the art in TDD can be found in (Hammond & Umphress, 2012). The purpose of TDD is to give the developers confidence that the code they produce works, as well as to guide the design of the code towards an easily testable structure. Additionally, the refactoring practice is also based on TDD to ensure that changes made to the code do not break any functionality (P. Abrahamsson et al., 2005).

In order to systematically observe the development process and to identify the artifacts created during it, we developed a prototype application, namely KnowLedge, for Android platform. The application intends to enable users learn and/or share knowledge in an interactive and social manner. Among others, the basic usage included functional requirements like browsing through categories to find existing knowledge on a topic or placing a request for a new explanation, instruction or tutorial, sharing knowledge in groups etc.

The overall system architecture comprises service oriented architecture, mobile application, remote database and usage of the global positioning system. In addition, as it can be seen in Fig. 2, the mobile application architecture is also intended to be multi-layered with three distinct but connected layers. The internal cohesion (see (Miller, 2008)) of the presented modules should be high, and at the same time the external coupling should be kept low.

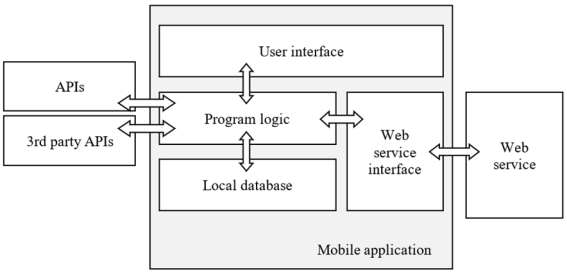

Figure 2. Architecture of KnowLedge application

The Mobile-D process with its clear technical specification was well documented and easy to follow and the overall development process took less time than initially planned. A few screenshots of the created application are visible in Fig. 3.

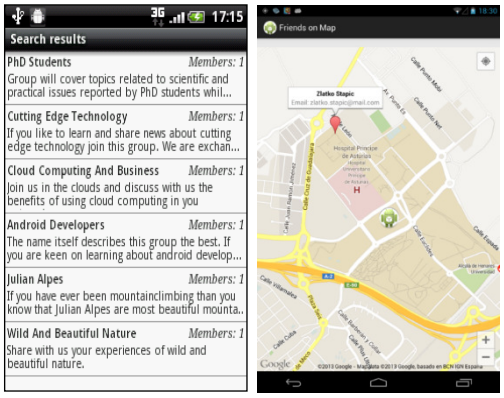

Figure 3. Screenshots of KnowLedge mobile application

3 Identification of artifacts

As there are many definitions of artifact (e.g. from Hilpinen (2011) or from Parker (2011)), we have adopted the definition from Conradi (2004) who says that artifact is "*any piece of software (i.e. models, descriptions, code) developed and used during software development and maintenance*". As the goal of this research was to analyse only the structural and semantic aspects of the sets of artifacts, we performed an analysis only from the semantic concept view, while other possible views, such as procedural concept view or pragmatic concept view are not covered by it. Thus, we only observed the artifacts and their connection to the activities and tasks as it can be seen in Fig. 4.

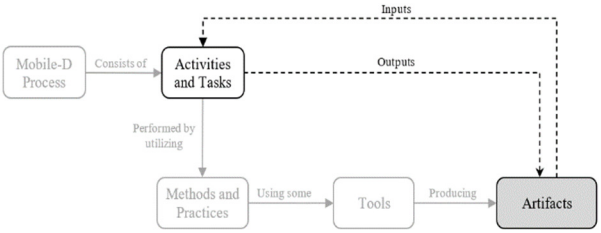

Figure 4. Focusing semantic of artifacts and their origin

We performed the artifacts analysis in two steps. Firstly, we analysed the Mobile-D process library (P. Abrahamsson et al., 2005) and identified the documents and other platform-independent deliverables at a high level of abstraction. Secondly, as the approach of identifying and grouping the artifacts only according to the phases of the origin would not be a good way, and as during the implementation phase we collected the additional data on the artifacts, we systematized and described all identified artifacts using the template presented in Table 1.

Thus, from the conceptual point of view, we created a solid basis for identifying not only the documents that had been created, but also other artifacts that might be hard to identify if the project was performed outside the laboratory.

Table 1 shows a part of the list of the identified artifacts, along with their initial classification, description and connection with the Mobile-D phases. We used standard CRU notation for denoting the artifacts that were created (C), used/read (R) and updated (U).

Table 1. An excerpt from the list of identified artifacts in development process for Android

Artifact name	**Type**	**Description**	**Phases inputs and outputs**									
			I		**II**		**III**		**IV**		**V**	
			Input	**Output**	**Input**	**Output**	**Input**	**Output**	**Input**	**Output**	**Input**	**Output**
Mobile-D process library	Document	Process library describing the Mobile-D methodology in detail. Used as methodology guidelines in every phase. (P. Abrahamsson et al., 2005)	R		R		R		R		R	
Product proposal	Document	Generated before the development process. Describes the initial and general idea on the product.	R									
Project plan	Document	Contains all information on project including definition of customer group, scope, planned activities and their duration, plans on documentation etc. Aligned with agile practices, this document is also updated during the iterations.		C	R	U	R	U				
…	…	…										

The identification process resulted in total of 60 different artifacts for Android development process, which are then analysed in order to see which of these artifacts are potentially reusable in development process for other platform such as iOS. These artifacts are grouped into 12 different categories as presented in the table below. The full list is available in appendix 1.

Table 2. Types of artifacts in Android development

Artifact type	Description
Document	Represents used documents or created artifacts that are published as documents during or at the end of development process.
Embedded document	Represents document that could be observed as stand-alone artifact, but is usually included in some other document.
Template	Represents templates that are used to create some artifacts.
Model	Represents models that are created during the development process. Models could be observed as stand-alone artifacts, but are usually presented as a part of some document.
Model element	Represents the atomic level (i.e. integral) artifact that could be observed as stand-alone and is used to create models.
Code	Represents any artifact created during the implementation and is written in any programming or description language.
3rd party example	Represents code artifacts created by third party and used as examples of implemented functionality or to solve some programming issue.
Software tool	Represents software tools used during the entire project.
License	Represents individual-specific unique key that is obtained or used during the development process.
Standard	Represents document containing formal and internationally recognized description of some concept or element.
Publishing resource	Represents resources that are created during the development process and are used in publishing purposes.
Product	Represents final product as most important project deliverable.

As the application was designed to cover the most common mobile application development use-cases (see Fig.4), it is not expected that the number of artifacts or their types would increase significantly if the general size of the application would be bigger. The KnowLedge application along with belonging tests and backend services has approximately 12.000 lines of code (LOC).

Some documents contain parts (document artifact) that should be observed separately which is why we identified them as a specific (new) type. Similarly, the model element could be observed as a stand-alone artifact used to build more complex models.

4 Reusability analysis

Mijač (2015) defines reusability as property of a software asset that indicates its probability of reuse, while the reuse is defined as the use of existing software and software knowledge to construct new software. In the context of the following analysis, we have adapted the mentioned definitions.

In the cross-platform reusability analysis we found that 50 artifacts (71.43% of Android identified artifacts) are mutual to both development cases. Additionally, many of these mutual artifacts are platform independent as being products of methodological approach. In total, 20 out of 50 identified mutual artifacts (40%) should be created or obtained only once, as these were identical in both development processes. On the other hand, there are 13 artifacts (26%) that could be partially reused while performing the development process for the second or any other target platform. Finally, we recognized 17 artifacts (34% of all common artifacts) with a very low level of possible reuse. They were classified as ones that should be developed from scratch for every target platform. A preview of results of the cross-platform analysis can be seen in Table 3. All other artifacts were classified as platform dependent artifacts, which also have some reusable semantic or syntactic parts like sequencing, iterations, algorithms etc. The full list of common artifacts, along with their classification is available in appendix 2.

Table 3. An excerpt from the list of mutual artifacts in Android and iOS case

Artifact name	Identical	Partially reusable	Different
Mobile-D process library	X		
Product proposal	X		
Initial requirements document	X		
Project plan		X	
Project plan checklist		X	
Project plan checklist template	X		
Measurement plan		X	
Architecture line description			X
…			

In total, 33 artifacts (66% of the mutual artifacts) are completely or partially reusable which encouraged us and provided a solid basis and motivation for the next phases of our research towards methodological interoperability and artifact reusability.

5 Conclusion

In this research we have presented the results of reusability analysis of methodological (i.e. process related) and development artifacts that arise in development of mobile applications for two or more target platforms. Firstly, we analysed the Mobile-D process library and identified the documents and other platform-independent deliverables at a high level of abstraction. Secondly, we collected the additional data on the artifacts and identified 60 different artifacts for Android development process which are grouped in 12 different categories. In the cross-platform reusability analysis that followed, we found that 71.43% of artifacts are common and present in all development processes in multi-platform development, while 66% of these are completely of partially reusable confirming that artifacts reusability presents solid basis for improvement in multi-platform mobile application development process.

In our further research we would like to semantically describe the methodology driven development process as well as the artifacts which arise in it, with the special focus on the reusable artifacts, in order to create a novel methodological framework for development of multi-platform native mobile applications.

Acknowledgments

The author would like to thank the reviewers for their valuable comments and suggestions that significantly improved this paper.

References

Abrahamsson, P, Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., & Salo, O. (2004). Mobile-D: An agile approach for mobile application development. Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications - OOPSLA '04, 174. https://doi.org/10.1145/1028664.1028736

Abrahamsson, P., Hanhineva, A., Hulkko, H., Jäälinoja, J., Komulainen, K., Korkala, M., Koskela, J., Kyllönen, P., & Eporwei, O. T. (2005). Agile Development of Embedded Systems: Mobile-D (Agile Deliverable D.2.3; p. 203). ITEA. http://www.agile-itea.org/public/deliverables/ITEA-AGILE-D2.3_v1.0.pdf

Adobe Inc. (2020). Adobe AIR | Deploy applications across platforms and devices. https://www.adobe.com/products/air.html

Agarwal, V., Goyal, S., Mittal, S., & Mukherjea, S. (2009). MobiVine: A middleware layer to handle fragmentation of platform interfaces for mobile applications. Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, 24:1–24:10. http://dl.acm.org/citation.cfm?id=1656980.1657013

Conradi, R. (2004). Software engineering mini glossary. http://www.idi.ntnu.no/grupper/su/publ/ese/se-defs.html

Facebook Inc. (2020). React Native · A framework for building native apps using React. https://reactnative.dev/

Google Inc. (2020). Flutter—Beautiful native apps in record time. https://flutter.dev/

Hammond, S., & Umphress, D. (2012). Test driven development. 158. https://doi.org/10.1145/2184512.2184550

Hilpinen, R. (2011). Artifact. Stanford Encyclopedia of Philosophy. http://plato.stanford.edu/entries/artifact/

Hosbond, J. H. (2005). Mobile Systems Development: Challenges, Implications and Issues. In J. Krogstie, K. Kautz, & D. Allen (Eds.), Mobile Information Systems II (Vol. 191, pp. 279–286). Springer Boston. http://dx.doi.org/10.1007/0-387-31166-1_20

La, H. J., & Kim, S. D. (2009). A service-based approach to developing Android Mobile Internet Device (MID) applications. 2009 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), 00(MID), 1–7. https://doi.org/10.1109/SOCA.2009.5410278

Mijač, M., & Stapić, Z. (2015). Reusability Metrics of Software Components: Survey. https://doi.org/10.13140/RG.2.1.3611.4642

Miller, J. (2008). Cohesion And Coupling. MSDN Magazine - The Microsoft Journal for Developers, 23(10). http://msdn.microsoft.com/en-us/magazine/cc947917.aspx

Parker, P. M. (2011). Definition of artifact. Webster's Online Dictionary. http://www.websters-online-dictionary.org/definitions/artifact

PhoneGap. (2020). Take the pain out of compiling mobile apps for multiple platforms. PhoneGap Build. https://build.phonegap.com/

Rahimian, V., & Ramsin, R. (2008). Designing an agile methodology for mobile software development: A hybrid method engineering approach. Proceedings of Second International Conference on Research Challenges in Information Science, RCIS (2008), 337–342. https://doi.org/10.1109/RCIS.2008.4632123

Shah, K., Sinha, H., & Mishra, P. (2019). Analysis of Cross-Platform Mobile App Development Tools. 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 1–7. https://doi.org/10.1109/I2CT45611.2019.9033872

Spataru, A. C. (2010). Agile Development Methods for Mobile Applications [PhD Thesis, University of Edinburgh, The University of Edinburgh]. http://www.inf.ed.ac.uk/publications/thesis/online/IM100767.pdf

Stapic, Z., Mijac, M., & Strahonja, V. (2016). Methodologies for development of mobile applications. 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 688–692. https://doi.org/10.1109/MIPRO.2016.7522228

Supan, D., Teković, K., Škalec, J., & Stapić, Z. (2013). Using Mobile-D methodology in development of mobile applications: Challenges and issues. Razvoj Poslovnih i Informatičkih Sustava CASE 25, 91–98.

VTT Technical Research Centre of Finland. (2006). Mobile-D Online Presentation (Web Application). AGILE Software Technologies Research Programme. http://agile.vtt.fi/mobiled.html

Appendix

1. Identified artifacts in development process for Android, available at: http://tiny.cc/identified_artifacts
2. Common artifacts in development for different platforms and their reusability classification, available at: http://tiny.cc/common_artifacts