Introducing A Microservice-Based Mobile Software Product Line – a Technical Debt Perspective

Luka Pavlič, Tilen Hliš, Marjan Heričko, Tina Beranič

Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, Maribor, Slovenia

{luka.pavlic, tilen.hlis, marjan.hericko, tina.beranic}@um.si

Abstract. *The software product lines (SPL) enable development teams to fully address a systematic reuse of a shared assets to deliver a family of a similar software products. Mobile applications are an obvious and proven candidate for employing an SPL approach. Technically speaking, the SPL implementation varies. In addition to the industry-accepted best practices, including build-time directives, advanced design patterns employment, custom builds etc., the microservice architectural style (MSA) can also be used in terms of realizing SPL. Our empirical data demonstrate the impact of a SPL approach in multiple edition mobile application development. The next step in our research is to verify what the impact of the SPL implementation with MSA in the mobile application area is. Our hypothesis is, that in addition to delivering high-quality features rapidly, the positive impact on the accumulated technical debt will be demonstrated as a direct result of employing MSA to mobile development.*

Keywords. Software development, reuse, microservice architecture, software product lines, mobile applications, technical debt

University of Maribor Press

DOI https://doi.org/10.18690/um.3.2022.29
ISBN 978-961-286-658-7

1 Introduction

Reuse is one of the fundamental disciplines in software engineering. Usually, a software does not result in a single version or edition, specially tailored to certain customers. The software product lines (SPL) [2] is an approach to reuse, employed in case where a family of products shares several common functionalities [1].

The origins of microservice architecture stem from the issue of web services, which do not support all the principles of service-oriented architecture. A new phenomenon has emerged in form of microservices, which are independently developed and operated, automatically deployed reusable components, that communicate via standard protocols. Increasing attention to microservice architecture and its many advantages, emphasizing low coupling, independence of individual services, and enabled scalability, which offers systems with better performance [6]. Although this approach was not initially created to support the SPL, we see it as a perfect match.

The technical debt metaphor, rooted in the financial world, captures the amount of work that development teams owe to the product [7]. Managed carefully, it can be used as an important tool while weighting trade-offs between the high product quality and the rapid delivery of crucial functionalities carefully.

As previous research demonstrate and this paper summarize, the SPL approach has many positive implications [5]. In this paper we lay the foundation to our future research, when we will investigate the impact of the microservices-based mobile SPL. The accumulated technical debt will be investigated in detail. Our hypothesis is, that because of the microservice design, the accumulated technical debt will be more favourable.

2 Mobile Software Product Lines

The software product lines (SPL) proved to be an adequate solution to reuse in special cases, when several software products share a majority of functionalities, while only a fraction of functionalities are edition specific [2]. According to the original SPL idea, development efforts are directed towards developing core assets, while product development is a process of aligning core assets into final products. Authors [2] also proposes several patterns and their variants, to be used for SPL-based development.

Besides reusing technical building blocks, these also include reusing procedures and rules, associated with the software. The SPL approach brings additional costs also: architecture, building blocks and individual tests should include the possibility of variability, while business plans must be made for multiple products, not just one. However, the long term claimed contributions of SPL are as follows [2]: up to 10x improved productivity, up to 10x improved quality, joint development costs reduced by up to 60%, shortened time-to-market by up to 98% and the possibility of moving to new markets is measured in months, not in years. Authors [1] define the SPL-approach as a tool to effectively cope with variabilities. The authors address three types of variabilities [3]:

- functionality presence,
- the lack of functionality, and
- a different functionality realization.

2.1 Our Approach to Manage Software Product Lines

First steps in our SPL implementation included requirements gathering, designing and testing planning for the functionalities, that were collected in a multi-dimensional table. Functionalities were not only listed, but also described in terms of which edition functionality was available and if and what specialities were required for a particular functionality in a particular edition. This is how development team ended with functionalities written in several categories: common (all editions), optional (only in selected editions) and alternative (edition-specific implementation of the same functionality). Please see the Table 1 for all different editions and their difference in terms of functionalities count.

Table 1. Final editions compared: functionality-based differences.

Edition	Based on	Base f.	Optional f.	Alternative f.	F. count (B+O)	F.diff (O+A)	Diff (%)
Pro	Free	45	8	2	53	10	19
Free	Core	45	0	3	45	3	7
Alpha	Core	45	9	5	54	14	26
Test	Pro	53	0	1	53	1	2
Demo	Free	45	8	2	53	10	19
BB Pro	Core	45	0	3	45	3	7
BB Free	Free	45	0	3	45	3	7

Figure 2 shows available assets (components implemented as Android libraries), from which 7+1(Core Module) are fully functional Android applications. A set of functionalities is present in particular application edition by appropriate library in edition. Functionality absence in achieved by not including the library. The alternative implementation is achieved by including library and overriding (a part) of its implementation by employing appropriate design pattern. These include the use of object-oriented design, proven design patterns, extensions, and component parameterization. Design patterns [4] are used heavily, especially: factory, abstract factory, factory method, bridge, bean, adapter and others.

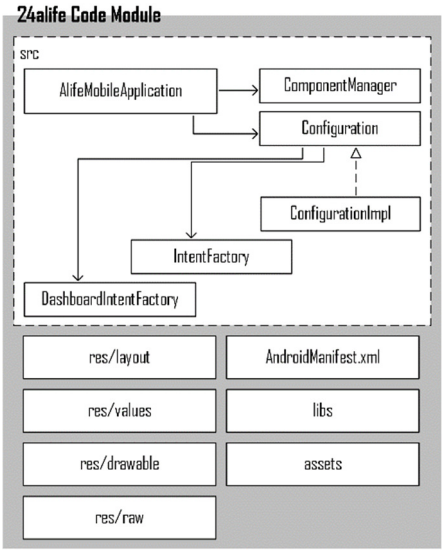

Figure 1. SPL architecture as a part of Core Module [5].

3 The Mobile Software Product Lines on Software Quality – Empirical Study Results

We will present our research outcomes, done during one year of development of mobile applications for two mobile platforms, Android and iOS, sharing a common set of functionalities [5]. Mobile applications are a part of a larger project which also included backend cloud solutions, a web portal, a media streaming server and tablet applications.

Several editions of mobile applications, applications for the Android platform, were managed with the introduction and implementation of the Software Product Line (SPL) approach, while other set of applications, applications for the iOS platform, were managed with more traditional methods of reuse, e.g. branches in version management system, sharing the same codebase, but compiling it several times, using compiler directives, runtime checking, etc. Android and iOS development teams shared the same set of functionalities that had to be developed and they were given the same time to finish the implementation. This industry-based setup gave us the opportunity to explore and share interesting pre-, mid- and post-development empirical data, compiled to research observations on SPL approach implications (see Figure 2).

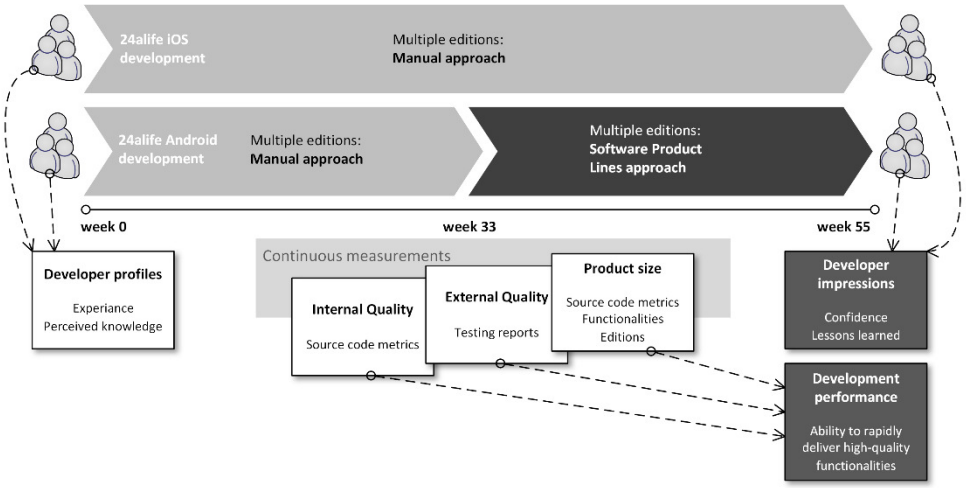

Figure 2. Gathering research empirical data during the development process [5].

The results of the research are as follows [5]:

- The SPL approach results in 126% higher functionality-based velocity (240% higher, compared to single-edition development).
- The SPL approach enabled developers to adopt an additional 100% of new functionalities across several editions with the same effort level.
- Internal quality is not affected by introducing SPL approach.
- SPL approach enhances external quality.
- Managing several editions using non-SPL approach would reduce external quality.
- The SPL approach have a positive impact on developers' confidence in delivering new functionalities and releases frequently.

4 Discussion and Future Work

Our research clearly demonstrates the benefits of employing the SPL approach to enhance a reuse in a family of a similar applications. In addition to the quality improvement, it enable development teams to deliver more valuable functionalities in shorter time.

The software product lines approach can be achieved using different approaches and technical implementation. Microservices seem a natural and a promising approach to do so. Our aim of further research in the field is to verify their application to handle variabilities. We primarily want to see, if and to what extend the impact would be. In addition to improved quality and accelerated functionality delivery, we believe that the application of microservices would also enhance internal quality and, consequently, lower accumulated technical debt in the family of similar products.

Acknowledgments

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

References

[1] Cavalcanti YC, Machado IC, Anselmo P. 2013. Handling variability and traceability over SPL disciplines, software product line–advanced topic, Edited by Abdelrahman Osman Elfaki, Rijeka, 2013.

[2] Clements P, Northrop L. 2001. Software product lines: practices and patterns. Boston: Addison-Wesley.

[3] Clements PC, Bachmann F. 2005. Variability in software product lines, product line practice initiative, 2005.

[4] Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Boston: Addison-Wesley, 1998.

[5] Pavlič, L, Beranič, T, Heričko, M. A product quality impacts of a mobile software product line : an empirical study. PeerJ computer science. 2021.

[6] Lewis. J. Fowler. M. Microservices: A definition of this new architectural term. https://martinfowler.com/articles/microservices.html [last accessed 04/2022]

[7] Cunningham. W. The wycash portfolio management system. in OOPSLA '92 - Experience Report, 1992.