BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

INTELLIGENT METHODS FOR INFERRING SOFTWARE ARCHITECTURES FROM MOBILE APPLICATIONS CODEBASES

PhD student: Dragoş Dobrean
Scientific supervisor: Prof. PhD Laura Dioşan

2021

Abstract

With over two-thirds of the world's population using them, smartphones have become our most personal devices. We use these devices for communicating with others, entertainment, analyzing our health, ordering food, exercising, finding a date, and so much more. Slowly they have even started to be a replacement for wallets, as they allow us to make payments, store digital tickets, and validate the identity of the owner. With the rise of the popularity of these devices and the hardware advancements, mobile applications have become one of the most commonly written pieces of software.

The software architecture of a mobile product, especially in the case of complex ones, which are maintained and developed for long periods (years), plays a major role in their success. By correctly implementing a certain architectural pattern to a mobile application project, developers ensure the testability, extensibility, and flexibility of those projects, which represent a major advantage in a fast pace, high-demand market. Since the architectural health of a mobile codebase represents an important factor in its success, this thesis studies different approaches in which architectural issues could be spotted and fixed before they could affect the product. The scope of this thesis is to pave the way for building a system capable of identifying architectural issues early, in the development phase, or Continuous Integration (CI) / Continuous Delivery (CD) pipelines, and to lay a strong foundation of knowledge that could also be applied to other platforms.

Each software architecture has a set of constraints between its composing parts. Presentational applications (such as mobile apps) predominantly use layered software architectures. While the architectural rules between the layers of a system can be easily defined, a more important problem is the inference of the architectural layers from a mobile codebase. To create a tool that highlights architectural issues automatically, we need a system that could infer the components from the codebase and categorize them in architectural layers without the need for human interaction.

The approaches presented in this thesis take advantage of the Software Development Kits (SDKs) used in building those kinds of applications. By using the information from those SDKs insightful data can be extracted regarding the types of the components in the codebase and the architectural layer they should reside in. In addition to the usage of information from SDKs, Machine Learning (ML) techniques are also involved in some of the introduced approaches for aiding the process of categorizing the components of a mobile codebase in architectural layers. Moreover, when designing the approaches, we've also focused on portability, all the presented approaches are platform-agnostic, and can easily be extended to other mobile or non-mobile platforms.

The research presented in this work is at the confluence of two domains, Software Engineering and Artificial Intelligence. The goal of this study is to advance the field of Software Engineering by improving the process of developing presentational software products. New discoveries are usually the byproduct of the interaction between different study fields, and we strongly believe that the Artificial Intelligence methods can be successfully applied to Software Engineering and both fields can be driven forward from this symbiosis. The problem of inferring software architectures from mobile codebases is heavily related to the field of Software Engineering. However, while developing approaches for enhancing this process we've tried to tackle every problem we encountered from an Artificial Intelligence perspective.

In this work, we have analyzed the most common architectural patterns used on mobile platforms, together with their advantages and disadvantages. We have discovered that the precursor of a large number of those architectural patterns, Model View Controller (MVC), is still one of the most used presentational patterns, especially in mobile codebases. The focus with this initial work was to study and find ways for automatically detecting the composition of the architectural layers of an MVC architecture from a mobile codebase.

Our first approach – Mobile software architecture checker system (*mACS*), the deterministic one, achieved good results in solving this problem, by using the information available in the mobile SDKs and a set of heuristics. Next, since there are other, more complex architectural patterns used in the mobile codebase, we have moved our focus to those, and try to come up with an approach that would allow us to detect other types of architectural layers, not only those present in the MVC. The second proposal – Clustering architectural layers (*CARL*) is the non-deterministic approach, in which machine learning techniques are used for splitting the components of a mobile codebase into architectural layers. This second approach was not as performant as our first one (the deterministic approach), but it had the extra flexibility that was lacking in the first attempt.

The third approach – Hybrid detection of architectural layers (*HyDe*) is a hybrid one, in which we have combined the deterministic approach with the non-deterministic one. This third attempt had an improved accuracy over the non-deterministic method and much better flexibility than the deterministic one and represents a good candidate for a solution that could analyze more specialized architectural patterns.

To sum up, this thesis managed to contribute to the field of software architecture in multiple research areas. This work shed some light on the importance of software architectures in an academic framework (students and instructors) as well as for practitioners. It analyzed and compared the most used architectural patterns on mobile platforms, and studied the most common mistakes of the most prolific one – the MVC. Moreover, with this work, we have formulated the problem of automatically identifying architectural layers from a mobile codebase mathematically and proposed three approaches (*mACS*, *CARL*, *HyDe*) for solving it. Furthermore, for testing the proposed approaches and comparing them, a benchmark of iOS and Android applications was constructed and the three approaches were compared from an intrinsic and extrinsic point of view and validated statistically with approach-specific metrics, and empirically using the interview method. In addition to this, we've also compared our detection methods with already existing ones from the literature by looking at how the information is used, how its analyzed, whats their granularity and flexibility. Our methods proved to have increased flexibility, meaning they can be applied to different platforms, and different programming languages with ease. Lastly, we have shown through experiments that the work presented in this thesis, can easily be ported to other mobile platforms, or even to completely different ones, with great results. With this work, we have opened a lot of research directions for future work. The results presented in this thesis, make us feel confident about achieving our goal – the creation of a mobile software architecture checker tool, that could be integrated into a CI/CD pipeline, or used locally, by developers or students. short-term research priorities revolve around improving the performance of our approaches. While the results obtained are promising, there is still a lot of work to be done, until we have an automatic solution that can detect architectural layers from mobile codebases. Our plans revolve around the software architecture tool, applying it to more platforms, and creating tools for improving the architectural knowledge of students.

Keywords: mobile applications software architecture; automatic static analysis; model view controller; automatic analysis of software architectures; structural and lexical information; software clustering; hybrid approach; software architecture detection; multi-platform

Contents