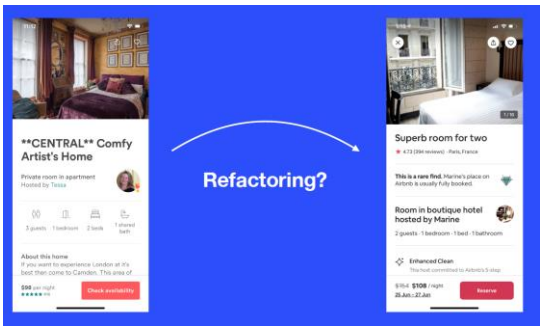
Intelligent methods for inferring software architectures from mobile applications codebases

Author: Dragoș Dobrean (under the supervision of prof. dr. Laura Dioșan)

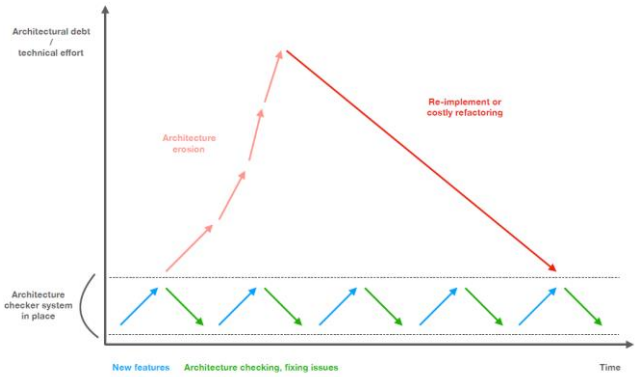
In software development refactoring periods are not viewed as productive tasks by the management team as in many cases, as quite often there are no visible changes to the product. One of the main reasons why refactoring is needed is because the software product has architectural issues which make the development of new features problematic. By identifying those issues early, when the code is still being developed, software projects could avoid long refactoring periods that are costly from both time and costs perspectives.



Mobile applications are one of the most written pieces of software nowadays, the software architectures used for building those products as well as their correct implementation represent an important factor in their success. By using the correct software architecture, mobile applications can easily be expanded to incorporate the latest software and hardware advancements and to quickly develop new features.

The aim of this thesis is to pave the way for building an automatic checker system for the software architectures of mobile applications, that can detect architectural issues, early in the development phase, before the code is pushed to production. For a checker system to detect architectural violations and inconsistencies, we need to know what the current architecture of the project we are analyzing is. With our study, we have laid the foundation of such a system by solving the problem of inferring software architectures from mobile codebases. We have developer three methods for inferring software architectures from mobile codebase:

- *mACS* -- a purely deterministic method that statically infers the architecture from a mobile codebase by using the information from the mobile Software Development Kits (SDKs) and a set of heuristics
- *CARL* -- a non-deterministic method that uses Machine Learning algorithms (clustering) -- which is more flexible than *mACS* (could be applied to more architectural patterns)
- *HyDe* -- a method that combines our two previous approaches (*mACS* and *CARL*), the goal of this method is to combine the adapatbility of *CARL* with the accuracy of *mACS*



We have validated our approaches on multiple mobile applications, from various domains, and we've found out that our methods have an average precision of 86% on all the analyzed codebases. In addition, we were also interested in the portability and extensibility of our methods to other mobile and non-mobile platforms. Our methods proved to have increased flexibility, meaning they can be applied to different platforms, and different programming languages with ease and great results.

With our work, we have paved the way for automatically detecting architectural issues in mobile and non-mobile codebases, by using a novel approach – leveraging the information from the SDKs and Machine Learning algorithms.