An analysis system for mobile applications MVC software architectures

Dragoş Dobrean, Laura Dioşan

July 20, 2019

Computer Science Department, Babes Bolyai University, Cluj Napoca, Romania

Table of contents

Introduction

Introduction

- Many companies built around their mobile applications (Instagram, Uber, Whatsapp, etc.)
- Projects need to be extensible, flexible, and to allow multiple people to work on them
- New software and hardware enhancements

Introduction

- Help the inexperienced developers write better code
- Validate weather or not some of the constraints of the architecture have been violated as early as possible in the development phase
- Insightful information for Management

Software Architecture Checker System

Software Architecture Checker System

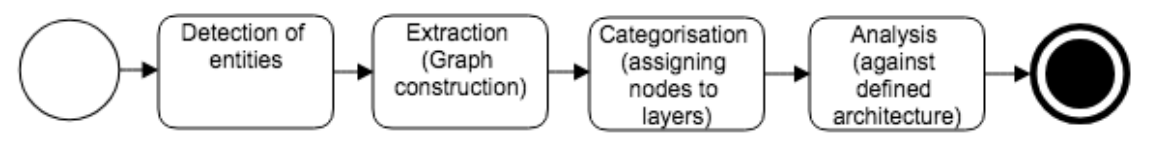

Figure 1: Software architecture checker system phases

- Works on Swift codebases
- Analysis MVC architectures
- Leverages SDK information for architecture extraction
- Automated process

Model View Controller

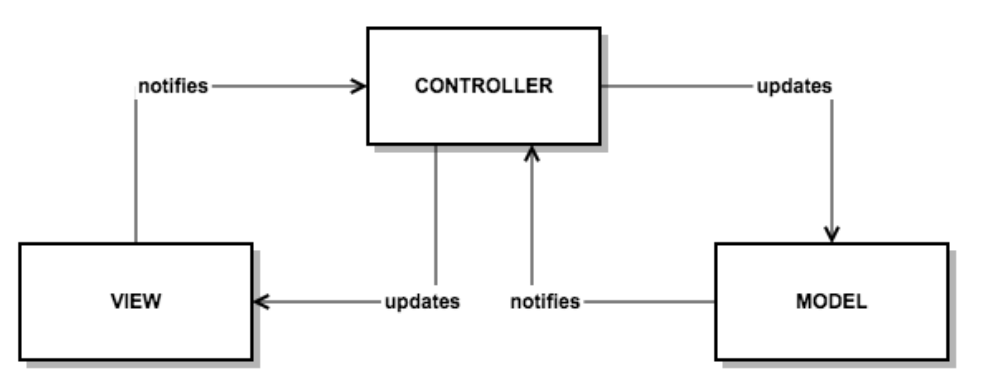

Figure 2: Apple's Model-View-Controller architectural overview

- One of the most used presentational patterns
- Heavily used on mobile platforms (iOS, Android, Windows Mobile)
- Separates the codebase in 3 layers **Model, View** and **Controller**
- "Father" of other architectural patterns — MVP, MVVM

Detection

- Components
  - **Classes**
  - **Structs**
  - **Protocols**
  - **Enums**
- Public and private **properties**
- Public and private class and instance **methods**

Extraction

It creates a dependency graph based on the information in that file, each node contains the following information:

- Name
- Type
- Inherited type
- Instance and class variables (name and type)
- Instance and class methods (together with parameters names and types)
- Path

Categorisation

Heuristics

$$Controller = \{n | pred_X(n, SDK'sController) = True, \text{ where } X \in \{instanceOf, inheritance\}\} \quad (1)$$

$$View = \{n | pred_X(n, SDK'sView) = True, \text{ where } X \in \{instanceOf, inheritance\}\} \quad (2)$$

$$Model = Components - View - Controller \quad (3)$$

Categorisation

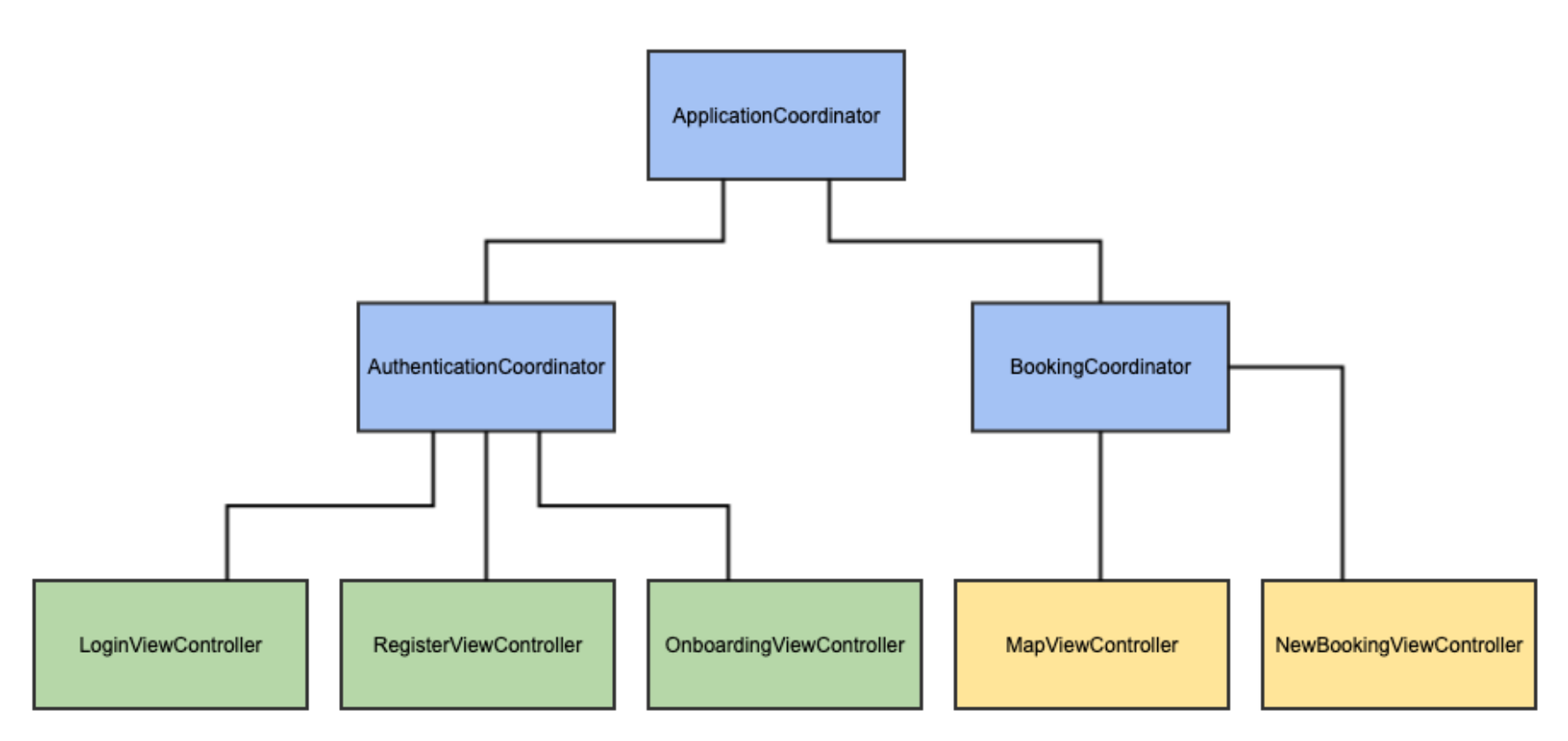

Figure 3: Coordinating controllers flows

Categorisation

Coordinating controllers

$$\begin{aligned} Coordinators = \{n | \exists v \in n.properties \text{ and } c \in Controller \\ \text{such as } pred_{instanceOf}(v, c) = True \text{ or } \exists m \in n.methods \\ \text{and } c \in Controller \text{ such as } pred_{using}(m, c) = True\} \end{aligned} \tag{4}$$

$$Controller = Controller \cup Coordinators \tag{5}$$

Analysis

Questions

Research questions

- RQ1 - How effective is the proposed categorisation method compared to manual inspections?
- RQ2 - What is the topological structure of mobile codebases using the proposed approach?
- RQ3 - Do mobile codebases respect the architectural rules?

Codebases

Application	Blank	Comment	Code	Source
Firefox	23392	18648	100111	open-source
Wikipedia	6933	1473	35640	open-source
Trust	4772	3809	23919	open-source
E-Commerce	7861	3169	20525	private
Game	839	331	2113	private

Table 1: Codebases size

Analysis

Methodology

- **Controller**: $L_{VC}^{CC} = \{l = (vc, cc) | vc \in \mathit{ViewControllers} \wedge cc \in \mathit{CoordinatingControllers}\} = \emptyset$ meaning that ViewControllers should not depend on other Coordinator controllers
- **View**:
  $L_{View}^{Other} = \{l = (v, o) | v \in \mathit{View} \wedge (o \in \mathit{Controller} \vee o \in \mathit{Model})\} = \emptyset$
  meaning that all components in the View layer should only depend on components within the same layer
- **Model**:
  $L_{Model}^{Other} = \{l = (m, o) | m \in \mathit{Model} \wedge (o \in \mathit{Controller} \vee o \in \mathit{View})\} = \emptyset$
  meaning that all components in the Model layer should only depend on components within the same layer

Analysis

Approaches

- MVC without Coordinating Controllers (SimpleCateg)
- MVC with Coordinating Controllers (CoordCateg)

Analysis - SimpleCateg

Layers

H_1 $Controllers \leftarrow CategorisationVCs(Comps.)$

H_2 $Views \leftarrow CategorisationViews(Comps. \setminus Controllers)$

H_3 $Models \leftarrow Comps. \setminus (Controllers \cup Views)$

Dependencies rules

R_1 $L^{Others}_{View} = \emptyset$ – all View components depend only on other View components

R_2 $L^{Others}_{Model} = \emptyset$ – all Model components depend only on other Model components

Analysis - CoordCateg

Layers

H_4 $Controllers \leftarrow CategorisationVCsAndCCs(Comps.)$

H_2 $Views \leftarrow CategorisationViews(Comps. \setminus Controllers)$

H_3 $Models \leftarrow Comps. \setminus (Controllers \cup Views)$

Dependencies rules

R_1 $L_{View}^{Others} = \emptyset$ – all View components depend only on other View components

R_2 $L_{Model}^{Others} = \emptyset$ – all Model components depend only on other Model components

R_3 $L_{VCs}^{CCs} = \emptyset$ – all ViewController components should not depend on Coordinator components

Evaluation

RQ1 - How effective is the proposed categorisation method compared to manual inspections?

- SimpleCateg - average accuracy of 89.6%
- CoordCateg - average accuracy of 86.2%
- Codebases using coordinators scored better than the others
- 100% accuracy on the simplest codebase

Evaluation

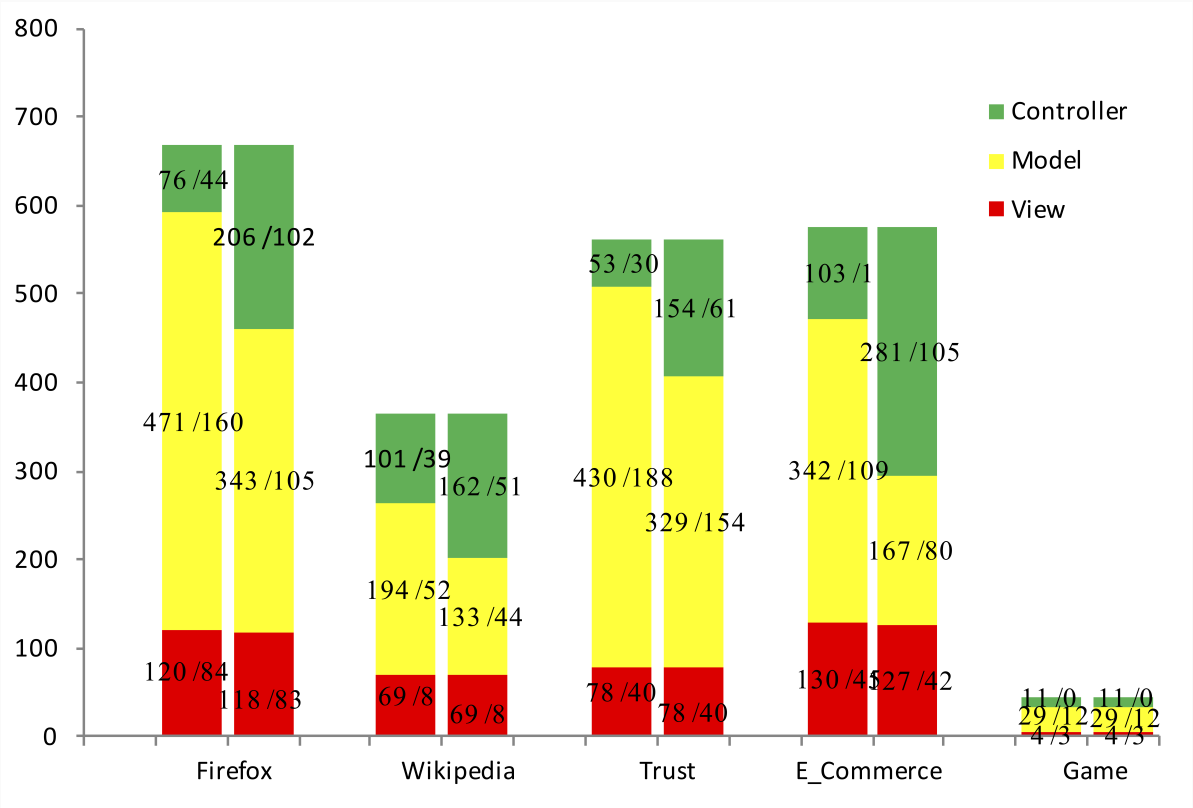

Figure 4: RQ2 - Topological structure of mobile codebases using the proposed approach

Evaluation

RQ3 - Do mobile codebases respect the architectural rules?

- R_1 and R_2 violated in both approaches
- R_3 violated in 3 of the 5 analysed codebases
- Model layer is the most problematic

Evaluation

Threats to validity

- Internal validity - mismatches between the purpose given by developer and the one obtained by inference, Swift only codebases
- External validity - the study was focused on MVC
- Conclusion validity - should be tested on more applications to strengthen our findings

Conclusions

Conclusions

- Issues come from bad implementation
- No architecture can enforce stop writing bad code
- Software Architecture Checker system adds another layer of architectural trust

Conclusions

- SDK information can be successfully used for inferring the architecture of a codebase
- Provides useful information for management and developers
- Adds another layer of architectural trust by integrating it in a CI/CD pipeline
- Can be generalised to other platforms where the products rely on an SDK

Further work

Further work

- Use AI & ML algorithms for improving the categorisation process
- Integrate the system into real CI/CD pipelines

Questions?