# Individual - Architecture Recovery

(Nicklas Jeppesen)
Course code: KSSOARC2KU

May 9, 2024

# Contents

# List of Figures

# Chapter 1

# Introduction

The goal for this project is create a Static Analysis of the code base: https://github.com/zeeguu/API. Where the goal should be to be able to take a given function and create a sequence diagram from the methods.

## 1.1 Real work life motivation

Why this project? why is this usefull and and why static analysis instead of dynamic analysis?

Motivation from this project, came by when I began my fulltime job where I had to maintain an existing relative large code base, where the code base was part of a eco-system, of multiple systems, where the system could get a request from another system and it, perform other request to other system, and to then return a response. There didnt exists documentation that explain the every single workflow, in this system, so to maintain and finding bugs became very hard. The idea was then, what is, a system, could cross systems create a sequence diagram, to virsualize the dependicy of sub system, even now micro-services and monolith distributed system has become more popular.

This report take the first step on this journey, and try to create a static analyzer which can analyze a single request method to create a sequence diagram from it.

### Why static analyze and not dynamic analyzer?

Its well known static analyzer have serveral limitation that danamic analyser dont have, for this it can be difficuilt to determine dead code, wheter it used or not, also if you have implementeted the factory patterns in your code, it

hard to say, if you have code that is never executed. So Dynamic code seems to be a better options. Now in my real work case. I had two issues to.

first, I was not able to run the program local on my machine, because it needed specific installations files and licence, it was not possible. Even then, we had serveres where the systeem was installed, but then the second problem occur. It needed specific knowledge to trigger a given request in the system, from another system, that I didnt had, and required a colleage. These are the motivation for using a static analyzer, that can analyse not running code, to do reverse engiernering to get an overview of the system. lastly, dynamic analyze is easier in some language than others like it easy in python, but more challenging in languene like java. Where statical analyze does not depend on built in function in the langues.

# Chapter 2

# Methodology

## 2.1 Tools

to creat this project. First this project used to provided from to colab pynb file. Thereafter it used the knowledge about dynamic analysis, and static and dynamic analyzer. To actual analyse the non running code, rich set of regex function was needed to be developered in the progress, for the following reason:

- Used to extract the imported modules in a file, first inspired by the provided code, but later modified.

- regex is also used to get a list of all functions definiton in a file, used later to figure out what module/file a function call in a function is comming from.

- Used to extract a function body from a string file, and a function name, to analyze.

- regex pattern matching is also used to get a list of function called in a function, for further analysis.

### 2.1.1 Creating the sequence diagram

For drawing the sequence diagram, I use a third party tool called mermaid-cli, my code create a string that this tool can transform into a sequence diagram.

# Chapter 3

# Results

## 3.1   what the code provide

By providing a filepath as string and function name to the system, it output a png file naming the provided function name containing the the sequence diagram of function. The sequence diagram module names as object, that is because something a file can import a function from a module and not a class, so safe choice is to name the object as the module, and then horientizal line as the function.

## 3.2   workflow

# Chapter 4

# Discussion

## General limitation of Static analys

## 4.1 Limitation for this codebase

sectionFurther development

# Chapter 5

# Time allocation

# Chapter 6

# Github

https://github.com/nicklasjeppesen/software_individual_report

# Chapter 7

# Appendix