An Analysis of Microservice Extraction Techniques

***Abstract*—The microservice-based architecture – a design principle wherein an application is divided into a suite of independently deployable services – has been frequently adopted by organizations looking to expand the scalability, deployability, and maintainability of their software systems. Despite the architecture's recent popularity, there is a lack of systematically-evaluated tools that automatically refactor monolithic applications into high-quality microservice-based systems. In this study, we select the most prominent approaches to automated microservice extraction — one that is based on static analysis and another that is based on dynamic analysis. We run these approaches on real-world monolithic applications, and evaluate the results against microservice-based representations of these applications produced manually by third-party experts. We discuss the accuracy and limitations of the considered approaches and provide suggestions for future research.**

I. Introduction

JR: Microservices are important.

As the size and complexity of modern, large-scale software applications continue to grow, an increasing number of organizations are adopting architectural strategies that better support the new scalability, deployability, and performance demands of their software systems. The microservice-based approach – an architectural stretegy that involves dividing applications into separate, independently-deployable components – is increasingly being used as a solution to these growing needs.

JR: Splitting monolith to microservies is challenging.

Despite the technique's growing popularity, there is a distinct lack of ubiquitous, standardized procudures to reference when splitting a monolithic application into microservices. Most proponents of this architectural stule cite domain-driven and business capability-centered design as optimal approaches to monolithic decomposition Lisa: INSERT CITATIONS HERE Lisa: INSERT INFO ABOUT WHAT THESE TECHNIQUES ARE ARE WHY THEY ARE CONSIDERED "OPTIMAL"; however, such approaches often require significant familiarity with the structure and inner-workings of both the software system in question and the organization that manages it. In addition, for particularly large applications, the decomposition process entails manually sorting hundreds, if not thousands, of files, classes, and/or methods into groups – a process that requires significant time and effort whilst being prone to the subjective architectural preferences of whomever is performing the decomposition.

JR: Tools exist. Based on static and dynamic analysis. Discuss the theoretical differences between static and dynamic analysis.

JR: Select one static and one dynamic tool. Describe the tools.

JR: Compare under the common setup. Discuss the setup and evaluation subjects.

JR: Results of the analysis.

JR: Suggestions for future research.

The remainder of the paper is structured as follows. Section II JR: TBD.

II. Background

In this section, we describe the microservice-based application architectural principle and reasons for its adoption.

JR: can copy from Yingying's paper

III. Evaluation Methodology

Our investigation is guided by the following research questions:

RQ1: XXX

RQ2: XXX

To answer these research questions, we ....

A. Tools

Describe the selected microservice extraction tools: Bunch and FoSci. How selected, why, and what they do.

B. Subjects

Apps: How selected, why, and what they do.

C. Metrics

How selected, why, and what they do.

IV. Results

In this section, we discuss our findings and threats to their validity.

A. RQ1: XXX

B. RQ2: XXX

C. Threats to Validity

Selection of tools, selection of subjects, manual interpretation of the results.

V. Discussion and Implications

Lessons learned and suggestions for future work.

VI. Related work

All related work [**?**].

VII. Conclusion

(Conclusion)

VIII. Acknowledgments

(Acknowledgements)