Exploring the Relationships between Software Architecture and Source Code

Fangchao Tian
State Key Lab of Software Engineering
School of Computer Science, Wuhan University
tianfangchao@whu.edu.cn

Peng Liang
State Key Lab of Software Engineering
School of Computer Science, Wuhan University
liangp@whu.edu.cn

Abstract—Software Architecture (SA) is introduced as high-level structures and the blueprint of a software system. Over the last decades, SA was broadly researched in academia and adopted in industry. SA design and code implementation are two essential and integral parts of the software development lifecycle. In recent years, there has been a research trend on investigating the relationships between SA and source code. Exploring such relationships is conducive to software system maintenance and evolution. The recent researches on the relationships between SA and code are mainly focused on the following questions: How to recover SA from code? How to automatically generate code from SA? How to detect architectural problems from code? But little research has been conducted on systematically analyzing, synthesizing, and using the relationships between SA and code. This PhD research proposes systematic methods with the purpose of summarizing, classifying, analyzing, identifying, and using the relationships between SA and code to effectively maintain and evolve software systems.

Keywords—software architecture; source code; relationships

I. INTRODUCTION

During the systems development life cycle, architectural level elements - modules, components, decisions, tactics, and patterns - are implemented and transformed to low-level source code [1]. Hence, analyzing and using the relationships between software architecture (SA) and code is necessary and beneficial to improve the maintenance and evolution of software systems. Currently, limited research focuses on systematically analyzing the relationships between SA and code. The main purpose of this PhD research is to systematically summarize and classify the relationships between SA and code. To further compare the views of industry and academia on the relationships between SA and code, we plan to conduct a survey to collect data from industrial practitioners. Moreover, our research is aimed at exploring specific relationships worthy of study between architecture and code. One such type of relationship between SA and code is the relationship between architectural smells and code smells. As two types of smells in software development [3], both code smells (“implementation structures that negatively affect system lifecycle properties” [4]) and architectural smells (“commonly used architectural decisions that negatively impact system lifecycle qualities” [4]) significantly impact system quality. Furthermore, code smells are closely related to architectural smells [4]. This research plans to further explore the relationship between architectural smells and code smells by conducting a case study and a controlled experiment.

II. RELATED WORK

Several researchers have analyzed architecture from the code level and studied the relationship between architecture and code. Fontana et al. evaluated architecture quality by exploiting code smell relations and co-occurrences [5]. Haitzer et al. have proposed a method to ensure consistency between architecture and source code during the evolution of a (large) system [6]. Christel et al. proposed a tool-supported approach to define and detect architectural tactics by analyzing different versions of the source code of the analyzed systems [7]. Konnerth et al. presented an approach to automatically recover the architecture of a system based on a software structure model from source code [8]. Although, several primary studies investigated the relationships between architecture and code, to the best of our knowledge, only one SLR on architecture and code was conducted [2], which focuses on traceability approaches between these two assets of software systems.

For one specific relationship between architecture smells and code smells, only few studies have researched this topic. Oizumi et al. study the relationship between code-anomaly (popularly known as code smells) agglomerations and architectural problems [9]. They conclude that code-anomaly agglomerations can manifest architectural problems. Bertran et al. analyze how to assess SA quality by exploiting code smells relations [10]. The result shows that code anomalies can be indicators of architectural degeneration. Due to the lack of a clear definition and classification of architectural smells, these studies cannot specify how architectural smells and code smells relate to each other.

III. RESEARCH QUESTIONS

The purpose of this research mainly covers three aspects: (1) to systematically summarize and classify the relationships between SA and code; (2) to propose approaches with dedicated tools that can help architects and developers analyze, synthesize, and use the relationships between SA and code.
code; and (3) to further investigate the relationship between architectural smells and code smells. We formulate the following research questions (RQs) accordingly:

- **RQ1:** What kinds of relationships have been identified and analyzed between SA and code in literature?
- **RQ2:** What kinds of relationships have been identified and analyzed between SA and code in practice?
- **RQ3:** How architectural smells and code smells relate to each other?
- **RQ4:** How can developers identify and use the relationship between architectural smells and code smells?
- **RQ5:** How can code smells support the detection of architectural smells?

### IV. RESEARCH METHODS

To obtain the answers of the aforementioned RQs, we outline and explain the proposed research methods in this section. We employ empirical research methods to answer the five RQs. For RQ1 and RQ3, we plan to perform an SLR. For RQ2 and RQ3, a survey in industry will be conducted. For RQ4, we will use case study as the research method. For the last RQ, controlled experiment is applied for answering it.

#### A. Systematic Literature Review

An SLR is a well-defined and rigorous method for identifying, evaluating, and interpreting all available research relevant to a specific research question, topic area, or the phenomenon of interest. By conducting the SLR, we aim to not only present all the relationships existed between SA and code (RQ1), but also to collect and analyze the approaches and tools to identify the relationships (RQ3).

#### B. Survey

A survey can be employed to collect quantitative or qualitative data about research questions from subjects, e.g., practitioners in industry. By conducting a survey, we plan to use interviews and questionnaires to collect and analyze: what kind of relationships between architecture and code are frequently concerned by practitioners (RQ2); how architecture smells are related to code smells (RQ3)? By comparing the results of RQ1 and RQ2, we can further investigate whether the research outcomes on the relationships between SA and code in academia align with that in industrial community.

#### C. Case Study

Case studies are aimed at investigating contemporary phenomena by gathering a clear chain of qualitative and quantitative evidence in its context. We can gain a deep understanding of the relationship between architectural smells and code smells (RQ4).

When conducting a case study, we follow the guidelines proposed by Runeson and Höst to formulate the protocol and processes of the case study. Herein, we plan to use face-to-face semi-structured interview as the methods for collecting data. We will invite some participants both from academia and industry to discuss and answer the questions (RQ4). The discussion will be recorded in a voice recorder. Lastly, we will transcribe the voice format into text for analyzing and coding the data using e.g., grounded theory.

### D. Controlled Experiment

Controlled experiments are widely used to test hypotheses for tentatively answering RQs. To verify whether code smells can help developers detect architectural smells, we will divide participants into a controlled group and experimental group. Both groups are required to identify architectural smells from a code base with specific metrics. In the experimental group, the code base is labeled with code smells, while the code base is not labeled in the controlled group. By comparing the efficiency and effectiveness of detecting architectural smells in both groups, we can answer RQ5.

### V. EXPECTED CONTRIBUTION

By conducting the PhD research, we expect to obtain the following research results: (1) a systematic analysis and category of the relationships between SA and code; (2) specific methods about how to identify the relationship between architecture smells and code smells; (3) techniques to detect architectural smells from code smells.

### REFERENCES


