A Replicated Experiment on Architecture Pattern Recommendation based on Quality Requirements

Xiuhong Chen, Wenqian Zhang, Peng Liang, Keqing He
State Key Lab of Software Engineering, Computer School, Wuhan University, Wuhan
{chenxh, liangp, hekening}@whu.edu.cn

Abstract—Applying software architecture patterns in architecture design is an important way to construct high quality design of software architecture. Meanwhile, the selection of architecture patterns has a close relationship to the quality requirements of software systems to be developed. Applying a specific architecture pattern can usually address several quality requirements, and further optimize architecture design. In this paper, we review existing recommendation methods of architecture patterns based on quality requirements, then analyze and verify the validity of a recommendation method through a replicated experiment in software engineering. The result of this work provides architects a verified method to select suitable architecture patterns based on quality requirements.

Keywords—software architecture; quality requirement; architecture pattern; architecture tactic; experimental replication

I. INTRODUCTION

Software Architecture Patterns (AP) as a set of reusable solutions of architecture design are used to deal with specific design issues under a given context, by which the functional requirements and quality requirements (QR) are satisfied in architecture patterns. Similar design issues exist in different contexts, and using architecture patterns can provide empirical and reusable knowledge for architecture design, and meantime improve the efficiency and quality of design. Architecture patterns have been extensively used in the industry [1].

The quality of architecture design has a direct impact on the ultimate quality of the entire system. Using architecture patterns depends on the extent to which architects are familiar with the patterns [2], as well as architects experiences of adapting different patterns to solve specific design problems. How to select appropriate architecture patterns using quality requirements is not an easy task [3], especially for a novice architect. With the increasing of various quality requirements [4], the architecture design process becomes more difficult because architects have to make a trade-off among various quality requirements [5]. Therefore, it is meaningful for architects to make decisions with recommended architecture patterns along with the rationale and the consideration of various quality requirements [6], and further improve systematic architecture design and reuse of using architecture patterns.

There is some existing work on recommendation of architecture patterns based on quality requirements. In [7], Harrison et al. proposed an approach to determining which patterns should be given priority to recommend through evaluating the degree of quality requirements. A quantitative assessment method is proposed in [8], to analyze the impact of changed quality properties on the selection of architecture patterns. Kassab et al. studied the impact of specific architecture patterns on non-functional requirements (i.e., quality requirements), and proposed an approach to decide the priorities of recommended patterns by system-related quality requirements [9]. All these work intends to provide architects reasons for which patterns should be prior to be selected from the perspective of quantitative assessment of requirements. Nevertheless, none of them validated the effectiveness of their approaches in practical architecture design. The application of architecture patterns is still an experience-based activity, rather than an engineered process. Note that not all software systems are designed by architecture patterns, in this paper we focus on the architecture design that is related to application and reuse of architecture patterns.

To this end, we select empirical design results from commonly-used architecture design and verify the effectiveness of recommendation of architecture patterns based on quality requirements through a replicated experiment, in order to provide architects a verified method to select suitable architecture patterns. The rest of this paper is organized as follows. Section II analyzes the existing recommendation methods of architecture patterns, and introduces the recommendation method of architecture patterns to be validated in this paper as well as the experimental questions. Section III describes the experiment process design based on the empirical approach of experimental replications. In Section IV, two cases are presented to explain the experiment, and the result is discussed in Section V. Section VI concludes the paper and identifies directions for future work.

II. RECOMMENDATION OF ARCHITECTURE PATTERNS

Some existing work has been done on recommendation of architecture patterns based on quality requirements. Bode and Riebisch proposed a quantitative method to evaluate how quality attributes affect the selection of architecture patterns [8]. Harrison and Avgeriou proposed an approach of pattern-based architecture review to evaluate the degree of realization of system quality attributes [7]. These work focuses on pattern recommendation, but not verification of its effectiveness in architecture design. Kassab et al. proposed a method that takes architecture tactics as an intermediary towards quantitatively assessing the impact of architecture patterns on quality requirements, and makes recommendation of architecture patterns based on the assessment result [6]. Xu and Liang proposed a cooperative coevolution approach to automate the recommendation of architectural pattern implementation [11]. But these works did not empirically verify the consistency between the calculated assessment results and the actual use of patterns. We conducted a replicated experiment to fill the gap.
III. Replicated Experiment

In this paper, a replicated experiment [10] is designed to validate the effectiveness of approach proposed in [9]. The objective is to provide architects a validated approach to recommend suitable architecture patterns based on quality requirements. The research question (RQ) is:

RQ: Whether the approach proposed in [9] is effective in practice or not. The effectiveness is measured by comparing the consistency between the calculated quantitative results and the impact value in actual use of patterns.

A. Experimental Replication

The design of our experiment follows the experimental replication guidelines proposed in [10], and the main steps are as follows:

1) Introduce the original experiment or the experiment to be verified, such as methods, material, and processes, etc;
2) Introduce experimental replications;
3) Execute the experimental replications;
4) Compare the result of experimental replications with original experiment;
5) Conclude the experimental replications.

Experimental replications are intended to improve the confidence of existing research results and to identify the remaining problems.

B. Method to be Verified

The method proposed in [9] is described as follows: The relationships between specific architecture patterns and quality requirements are obtained by a matrix operation, and architecture tactics play an important and intermediate role in the matrix operation.

We uses matrix A to represent the relationships between architecture tactics and architecture patterns. Each element in matrix A represents the impact of a specific architecture pattern on an architecture tactic, i.e., the impact of using a specific architecture pattern on the adoption of an architecture tactic. Matrix B represents the relationship value between quality requirements and architecture tactics. Each element in matrix B represents the impact of specific architecture tactic on a quality requirement.

Using Formula (1) to get the resulting matrix C:

\[
C = \frac{A* B}{\text{the number of architecture tactics}}
\]

which shows the quantitative impact between specific architecture patterns and quality requirements. The examples of matrix A, B, and C will be provided with two case studies in Section IV.

Note that the element values in both matrix A and B are given by architects. "2" represents that a pattern can easily realize the architectural tactic without modifications. "1" denotes that it is relatively easy for the pattern to implement the architectural tactic with slight modifications. "0" shows that the pattern has no effect on the architectural tactic. "-1" represents that it is not easy for the pattern to implement the architectural tactic, and slight modifications are needed in order to reduce the negative impact on tactic implementation. "-2" denotes that it is not easy for the pattern to implement the architectural tactic; considerable modifications are needed to alleviate the negative impact on tactic implementation. Matrix M defined in Section IV to reflect the effect of specific architecture patterns on three quality requirements: Performance, Reliability, and Security. In different design contexts, the weight (importance) of each quality requirement is different. In this paper we treat each quality requirement as equally important.

C. Experiment Materials

Four commonly used quality requirements of architecture design, i.e., Performance, Reliability, Security and Testability, are taken into account. Meanwhile, four architecture patterns, i.e., Layer, Pipes/Filters, Broker, and Model-View-Controller (MVC), are considered as candidate patterns to satisfy above quality requirements. Note that, even though this work does not discuss functional requirements, it doesn’t mean that use of architecture patterns should not consider them. In this work we only focus on the recommendation of architecture patterns based on quality requirements.

TABLE I. THE RELATIONSHIP TABLE BETWEEN QR AND AP

<table>
<thead>
<tr>
<th>QR</th>
<th>Pipes/Filters</th>
<th>Layer</th>
<th>MVC</th>
<th>Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Reliability</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Security</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>Testability</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

According to the description in [1] and authors’ experiences, we quantify the impact of four quality requirements (QR) on architecture patterns (AP) in TABLE I, in which the meaning of the values is presented in Section III. TABLE II shows the impact of several architecture tactics on the implementation of reliability quality requirement, which is matrix B defined in Section III. Similarly, the impact of four architecture patterns (i.e., Pipes/Filter, Layer, MVC, and Broker) on architecture tactics for reliability are shown in TABLE III, which is matrix A in Section III.

TABLE II. THE IMPACT OF SEVERAL ARCHITECTURE TACTICS ON THE IMPLEMENTATION OF RELIABILITY QUALITY REQUIREMENT

<table>
<thead>
<tr>
<th>QR</th>
<th>Architecture tactic</th>
<th>Reliability</th>
<th>Performance</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault detection</td>
<td>Ping/Echo</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>Heartbeat</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>Exception</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>Voting</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Active redundancy</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Passive redundancy</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Spare</td>
<td>1</td>
<td>-1</td>
<td>0</td>
</tr>
</tbody>
</table>
TABLE III. THE IMPACT OF AP ON ARCHITECTURE TACTICS FOR RELIABILITY

<table>
<thead>
<tr>
<th>QR</th>
<th>Architecture tactics</th>
<th>Pipes/Filters</th>
<th>Layer</th>
<th>MVC</th>
<th>Broker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault detection</td>
<td>Ping/Echo</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>Heartbeat</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>Exception</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Fault recovery</td>
<td>Voting</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Active redundancy</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>Passive redundancy</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>Spare</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>Shadow</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>State resynchronization</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Rollback</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Fault prevention</td>
<td>Removal from service</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Transactions</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Process monitor</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

D. Experiment Design and Process

We first calculate the impact of specific architecture pattern on four quality requirements, i.e., Performance, Reliability, Security, and Testability, using the approach proposed in [9]. The result obtained is matrix C mentioned in Section III. We validate the effectiveness of the approach by comparing the values in TABLE I, when considering only a single quality requirement.

Then, we simultaneously calculate the impact of specific architecture patterns on two quality requirements: Performance and Reliability. This process shows the comprehensive impact of specific architecture patterns on Performance and Reliability quality requirements.

Finally, we simultaneously calculate the impact of specific architecture patterns on three quality requirements, namely matrix M in Section IV. This step shows the comprehensive impact of specific architecture patterns on three quality requirements Performance, Reliability, and Security.

The experiment design is shown in Fig. 1: First, the impact of specific architecture patterns on four quality requirements is calculated. If the calculation results (i.e., values in matrix C) are consistent with TABLE I, the experiment will continue, else the experiment process terminates, returning invalid (i.e., the proposed method is invalid).

![Fig. 1. The overview of experiment design](image)

E. Limitations of Experiment

There are several limitations of experimental replications used to verify the validity of a method. The possible limitations that may affect the experiment results are summarized as follows:

- Completeness of architecture tactics;
- Representation of impact value;
- Computational errors.

We will not delve into a detailed discussion of these limitations, as it is not the focus of this paper.

IV. CASE STUDIES

In this section, we use two representative cases to perform the replicated experiments. In case one, only the impact of specific architecture patterns on one quality requirement Reliability is calculated; in case two, the comprehensive impact of specific architecture patterns on three quality requirements is calculated. The calculated results are then compared with actual impact of architecture patterns to quality requirements, and the effectiveness of the verified method will be determined by whether the calculated impact and actual impact are consistent or not.

A. Case One

Applying the method proposed in [9] to calculate the impact of specific architecture pattern on one quality requirement Reliability. Matrix B is acquired with TABLE...
\[ B = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T \]

where \( T \) represents a matrix transpose, and \( A \) is:

\[ A = \begin{bmatrix} -1 & -1 & 1 & 0 & -2 & -1 & -1 & 0 & -1 & 1 & -1 & 1 & 1 & 1 \end{bmatrix} \]

Then we use Formula (1) to get matrix \( C \),

\[ C = A \cdot B / 13 = \begin{bmatrix} -0.462 \\ -0.615 \\ -0.462 \\ -0.385 \end{bmatrix} \]

Finally, the effectiveness of the method in [9] is determined by comparing the calculation results in matrix \( C \) with actual values.

### B. Case Two

In this case, we focus on the impact of specific architecture pattern on three quality requirements Performance, Reliability, and Security simultaneously. When taking Reliability as an example, one can get matrices \( D, E \) with the method in [9] and the impact value in TABLE I.

\[ D = \begin{bmatrix} -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & 1 & 0 & -2 & -1 & -1 & 0 & -1 & 1 & -1 & 1 & 1 & 1 \end{bmatrix} \]

\[ E = \begin{bmatrix} -1 & -1 & 1 & 0 & -2 & -1 & -1 & 0 & -1 & 1 & -1 & 1 & 1 & 1 \end{bmatrix} \]

\[ F = \begin{bmatrix} 0 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \]

\[ G = \begin{bmatrix} -1 & -1 & -1 & -1 & 1 & 2 & -2 & 0 & -1 \\ -1 & -1 & -1 & -1 & 1 & 0 & -1 & 1 & 0 & 1 \\ -1 & -1 & -2 & -1 & -1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \]

\[ P = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \end{bmatrix} \]

\[ Q = \begin{bmatrix} -2 & -2 & 0 & -2 & -2 & 1 & 2 & -1 & -1 \\ -1 & -2 & 0 & -2 & -2 & 1 & 2 & -1 & 0 \\ 0 & -2 & 0 & -2 & -2 & 1 & 2 & -1 & 0 \end{bmatrix} \]

The other four matrices \( F, G, P, \) and \( Q \), are obtained by referring to the impact values in TABLE II and TABLE III respectively. Using the six matrices obtained to calculate matrix \( M \),

\[ M = (E \cdot D / 13 + G \cdot F / 10 + Q \cdot P \cdot 9) / 3 = \]

\[ \begin{bmatrix} -0.273 & 0.450 & -0.116 \\ -0.320 & 0.231 & 0.011 \\ -0.184 & 0.621 & -0.330 \\ 0.057 & 0.095 & -0.034 \end{bmatrix} \]

From the values in matrix \( M \), we can get the comprehensive impact of specific architecture patterns on the three quality requirements Performance, Reliability, and Security. Note that the order of calculation of these three quality requirements does not affect the final calculation results.

### V. Experiment Results

The values in the second column of TABLE IV represent the impact of specific architecture patterns on Reliability. Compared with the values in TABLE I, the differences are shown in FIG. 2. Similarly, the impact of specific architecture patterns on Performance, Security, and Testability are shown in the other columns of TABLE IV and the comparison results are shown in FIG. 3, FIG. 4, and FIG. 5. With the comparisons, we can conclude that the results obtained by calculation using the method in [9] are consistent with the actual use of patterns according to architects' experiences.

<table>
<thead>
<tr>
<th>QR</th>
<th>Reliability</th>
<th>Performance (Efficiency)</th>
<th>Security (Integrity)</th>
<th>Testability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipes/Filter</td>
<td>-0.462</td>
<td>-0.300</td>
<td>-0.778</td>
<td>0.75</td>
</tr>
<tr>
<td>Layer</td>
<td>-0.615</td>
<td>-0.400</td>
<td>-0.222</td>
<td>0.75</td>
</tr>
<tr>
<td>MVC</td>
<td>-0.462</td>
<td>0.700</td>
<td>-0.889</td>
<td>1</td>
</tr>
<tr>
<td>Broker</td>
<td>-0.385</td>
<td>-0.100</td>
<td>-0.333</td>
<td>-0.25</td>
</tr>
</tbody>
</table>

![Fig. 2](image)

**Fig. 2.** The impact of AP on Reliability QR compared with the value in TABLE I

![Fig. 3](image)

**Fig. 3.** The impact of AP on Performance QR compared with the value in TABLE I
Then, the impact of specific architecture patterns on two quality requirements: Reliability, Performance and on three quality requirements: Reliability, Performance, and Security are investigated (see TABLE V and VI, respectively). One can find that the extent of the impact is varied, when different numbers of quality requirements are considered. For example, when considering the quality requirements Reliability and Performance, the impact of Broker pattern on Reliability is negative in TABLE V However, once Security is added as shown in TABLE VI, the impact value becomes positive.

The replicated experiment result provides architects with verified selection method of architecture patterns based on quality requirements. With the help of pattern recommendation and its rationales, architects can better balance pros and cons of different architecture patterns in realizing specific quality requirements. Furthermore, it could offer architects the latest recommendations of architecture design and assist architects to achieve the architecture design systematically, when quality requirements or priorities of requirements are changed.

![Fig. 4. The impact of AP on Security QR compared with the value in TABLE I](image1)

![Fig. 5. The impact of AP on Testability QR compared with the value in TABLE I](image2)

**TABLE VI. THE COMPREHENSIVE IMPACT OF ARCHITECTURE PATTERNS ON RELIABILITY, PERFORMANCE, AND SECURITY**

<table>
<thead>
<tr>
<th>AP</th>
<th>QR</th>
<th>Reliability</th>
<th>Performance (Efficiency)</th>
<th>Security (Integrity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipes/Filters</td>
<td>0.273</td>
<td>0.450</td>
<td>-0.116</td>
<td></td>
</tr>
<tr>
<td>Layer</td>
<td>-0.320</td>
<td>0.231</td>
<td>0.011</td>
<td></td>
</tr>
<tr>
<td>MVC</td>
<td>-0.184</td>
<td>0.621</td>
<td>-0.330</td>
<td></td>
</tr>
<tr>
<td>Broker</td>
<td>0.057</td>
<td>0.095</td>
<td>-0.034</td>
<td></td>
</tr>
</tbody>
</table>

**VI. CONCLUSIONS AND FUTURE WORK**

In this paper, we review existing architecture patterns recommendation methods based on quality requirements. Four commonly-considered quality requirements are used to verify the validity of the method in [9] through an empirical approach of experimental replications in software engineering.

The contributions of this work: (1) validate the effectiveness of an architecture patterns recommendation method through experimental replications. (2) provide architects a verified recommendation method of architecture patterns.

We outline our future work in two points: (1) propose an easily quantifiable and objective assessment method to measure the relationships between quality requirements and architecture patterns; (2) provide architects more experienced support to select appropriate architecture patterns through further considering the quality requirements used in practice.

**REFERENCES**


