Toward a Framework for Evaluating Heterogeneous Architecture Styles

Shahrouz Moaven\textsuperscript{1}, Ali Kamandi\textsuperscript{1}, Jafar Habibi\textsuperscript{2}, Hamed Ahmadi\textsuperscript{3}

\textsuperscript{1,2}Computer Department, Sharif University of Technology, \textsuperscript{3}Member of Young Researchers Club

Abstract

Evaluating architectures and choosing the correct one is a critical issue in software engineering domain, in accordance with extremely extension of architecture-driven designs. In the first years of defining architecture styles, some special quality attributes were introduced as their basic attributes. After a moment, by utilizing them in practice, some results were obtained confirming some of attributes; some others meanwhile were not witnessed. As software architecture construction process is dependent on and addressed by both usage conditions and quality attributes, in this paper a framework has been proposed to provide an environment and a platform that can cover evaluation of architecture styles with a technique that not only exploits both qualitative and quantitative information but also considering users' needs is possible precisely and with high quality. Moreover, we define a classification and notation in order to describe heterogeneous architectures. It provides us with the ability of evaluating heterogeneous architecture styles of a software system.

1. Introduction

Deciding about and selecting the suitable architecture for the system and its homogeneous subsystems is an indispensable parameter while the software system is being developed [1]. There are many parameters effecting the selection and its steps.

One of the ways to design software systems and guarantee their individual quality attributes is making use of architecture styles [2]. The effect of quality attributes of each architecture style might vary in different domains; i.e. they may have good or bad effect on different domains. In spite of existence of some attributes for each style in different texts, they steel suffer from the lack of discipline and integration in them; therefore, understanding the level to which benefits and disadvantages of quality and quantity attributes of architecture are considered is somehow impossible [3]. That makes difficult the comparison among capabilities and benefits of architecture. Moreover, because of obtaining the capabilities of software architecture styles with respect to a special domain in some cases, there is steel a sensible need for refinement and improvement of software architectures by virtue of developers' experiments. Consequently, deciding about the architecture style that should be selected depends directly on the realization of the first creator of software. In order to avoid making decision base on software developers' realization and conception, it is important to quantify knowledge of developers and compare them with candidate software architectures of a system. It should work as a complement to intuitive judgment while selecting architecture.

Due to the complexity and expansion of system domains, most of systems are composed of different architectural styles [4]. For example, the internal structure of a system component may have an architecture which is different from the architecture by which the component has been organized.

The goal of this article is to represent a framework for evaluating heterogeneous architecture styles. It provides us with: 1) obtaining the integrity and discipline needed to represent attributes of architecture styles, 2) defining investigated attributes of each style in accordance with its usage domain, 3) comparing styles under an equivalent comparison platform. Thanks to the flexibility of the proposed framework, the obtained results are suitable for the special domain. Furthermore, practitioners can follow the framework and obtain acceptable results without being forced to have special levels of experience.

In the following we discuss different evaluation approaches. Then we classify the current compositions and define a notation to overcome the complexities of heterogeneous architectures and to become able to represent them explicitly. The overall structure and platform for comparing different styles and describing its parts is present base on the defined notation.

2. Traditional approaches of architecture evaluation

There are three phases in traditional software architecture evaluation approaches: early, middle, and post-deployment [6]. The earliest evaluation phase happens after adopting initial high-level and high-priority architecture decisions; so we can evaluate initial decisions and detect poor choices. The middle evaluation phase occurs after obtaining some details about architecture design. The evaluation can occur at any point while details are obtained; so we can find problems of architecture design. Post-deployment evaluation phase occurs when system design is complete. At this point, both architecture and system exist; so we can answer the
question about whether or not the architecture matches the system implementation.

2.1. Evaluation and measurement methods of traditional approach

There are different methods and structures in order to evaluate architectures each one follows special goals and focuses on them. As a consequence, personal translations and interpretations should be interfered in these descriptions in order to perform a comparison and take advantage of them.

Generally, we can classify current evaluation methods into two categories: quantitative and qualitative. To use special software architecture, scientific texts typically provide us with some guidelines by which qualitative attributes could be concluded [7]. While a quantitative description is a mean to compare the level of satisfying different quality attributes in different domain spaces. In other words, quantitative evaluations represent the importance level of described benefits and capabilities.

2.1.1. Qualitative evaluation methods

Qualitative evaluation of architecture styles is in fact the combination of each architecture style with the special quality attributes which exist in them inherently. Questioning techniques are the most common methods that can be used to evaluate architecture with every quality. Measuring techniques can be used to answer special questions and address special software qualities (e.g. performance or scalability); they are not as extensive as questioning techniques.

There exist three kinds of questioning techniques named scenario, questionnaire, and checklist. They have some differences in applicability; but have same goal that is increasing the understanding about the amount of fitness between architecture and requirements.

- Scenario: Quality attributes (e.g. maintainability, security, performance, and reliability) become meaningful in the context. Scenarios are descriptive tools that are used to evaluate quality attributes in a context.
- Questionnaire: A list of general and relatively open questions that is applicable to all architectures [6]. Some questions are about architecture creation and documentation and some of them focus on descriptive details of architecture.
- Checklist: A set of very detailed questions that have obtained after having several experiences about some similar domains. They focus on special quality attributes of a system. Checklist is helpful to keep a balanced focus on all areas of the system [8].

2.1.2. Quantitative evaluation methods

Quantity comparison is performed to create a schema based on quality attributes. In scientific texts, quality attributes are found based on other information resources coming with an individual background and an experimental collection [3,7].

We should mention that each kind of comparison between quantitative frameworks and qualitative scientific texts has been performed with respect to topical interpretation of information exploited from scientific texts.

There are some cases in scientific texts in which a special quality attribute has neither been mentioned as a benefit nor as a capability of architecture.

Measuring techniques lead into quantitative results. These techniques answer the questions of measurement team about quality attributes of architecture instead of providing us with ways to create questions. These techniques are more completed and mature than questioning techniques.

- Metrics: Metrics are quantitative interpretations placed on particular observable measurements on the architecture, such as fan-in/fan-out of components [12]. With measuring techniques, the evaluation should focus not only on the results of the metric, but also on the assumptions under which the technique was used [6].
- Simulations, prototypes, and experiments: Constructing a prototype or performing a simulation of a system may help to establish and clarify architecture, but is mostly expensive. In other words, they are often a part of development process.

2.2. Comparing evaluation techniques

We should generally answer two kinds of questions in an evaluation. First one is about the architecture as an artifact (architecture evaluation). In the second, the focus is on the role of architecture in development process. It provides us with subjects like: how the architecture can be applied into the product lifecycle, whether or not it is profitable, and who is responsible for that.

Questionnaires and checklists can evaluate both aspects based on their questions. Scenarios, metrics, and prototypes evaluate architecture only from product side. Scenarios may focus on the role in some ways, but it is not their actual intent. Moreover, we compare them based on two other aspects: generality and level of details.

- Generality: questionnaire is a general-purpose technique that focuses on general-purpose issues and is applicable in any architecture [9]. Domain-specific techniques (e.g. checklist) focus on domain-specific issues. Scenarios are system-specific, but it is also possible to choose a scenario for a special issue. Metrics are often general-purpose, although there may be some domain-specific metrics. Simulations, prototypes, and experiments are mostly domain-specific, although there are some general purpose tools to construct simulation.
- Level of details: it indicates the amount of information that we need about architecture in order to perform an evaluation. Here we can understand when to use evaluation technique in the development
cycle. Coarse-grained approaches can be applied early in the design process, while fine-grained approaches must be applied when more decisions have been adopted [6].

3. Related work

From the works represented after ATAM, we can mention Attribute-Based Architecture Styles (or ABAS) which is an extension of concepts propounded in each architecture style [7]. In ABAS analytical models of quality attributes have been attached to every architecture style to confirm the correctness of performed design. In other words, it contains some of analytical models that let designers to predict their behaviors with respect to some quality attributes. ABAS provide designers with a kind of mapping between architecture structures and quality attributes.

In order to compare quality and quantity attributes, some works have been done. In [3], in order to represent quality attributes, on it, a questionnaire to eight specialists in software engineering. They filled in the questionnaire based on a method called Analytical Hierarchical Process in order to construct a quantitative framework of benefits and capabilities of different architecture candidates with respect to a set of qualitative attributes.

The quantitative approach represented in this article, enables us to compare architecture candidates with respect to different sets of desirable quality attributes. The architecture candidates and quality attributes can differ whenever the process runes. The pair wise comparison of those attributes is done by using the hierarchical comparative process of AHP [10].

Table 1. (FAS) [3]

<table>
<thead>
<tr>
<th></th>
<th>Microkernel</th>
<th>Blackboard</th>
<th>Layered</th>
<th>Model-View-Controller</th>
<th>Pipes and Fibers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>0.181</td>
<td>0.143</td>
<td>0.0565</td>
<td>0.0557</td>
<td>0.218</td>
</tr>
<tr>
<td>Functionality</td>
<td>0.319</td>
<td>0.121</td>
<td>0.225</td>
<td>0.113</td>
<td>0.153</td>
</tr>
<tr>
<td>Usability</td>
<td>0.106</td>
<td>0.127</td>
<td>0.255</td>
<td>0.104</td>
<td>0.0818</td>
</tr>
<tr>
<td>Reliability</td>
<td>0.122</td>
<td>0.4732</td>
<td>0.0930</td>
<td>0.105</td>
<td>0.144</td>
</tr>
<tr>
<td>Maintainability</td>
<td>0.183</td>
<td>0.275</td>
<td>0.221</td>
<td>0.390</td>
<td>0.271</td>
</tr>
<tr>
<td>Parsimony</td>
<td>0.109</td>
<td>0.557</td>
<td>0.130</td>
<td>0.120</td>
<td>0.115</td>
</tr>
</tbody>
</table>

4. Suitable method and platform for evaluating styles

Since 1994 in which a primary classification of architecture styles were created [2] and some structural and quality attributes were listed for them, the architecture styles have been applied in lots of software projects. Most of the times, the experimental results were matched with the listed attributes, but sometimes they have shown different results in practice. In some cases, either the results obtained in special application domains were different from the listed attributes or the attributes did not appear completely and comprehensively and the satisfaction level was different with respect to the working situation and platform [11, 12]. Some new results have obtained, even in the fields in which the designer was silent. These results are complements to the results listed by designers. Evaluating architecture styles without paying attention to their usage background squanders some attributes and benefits of architecture styles. Evaluating with respect to the usage background of styles can provide us with numbers in a parametric and quantitative way and we can decide with respect to them to evaluate styles in special domains [3, 15]. So, a framework which exploits both the quality attributes propounded by designers and the quantitative results explored from their background provide us with high precision and reliable evaluation results.

In the existing architecture evaluation methods, architecture styles are evaluated like architectures without paying attention to the attributes of architecture styles [13]. There is no method which is specific for evaluating architecture styles and exploits the beneficial attributes of architecture styles. Consequently, in this article, a structure and a technique is represented with the aim of documenting and formalizing architecture styles with respect to their quality and quantity attributes. Moreover, we try to describe heterogeneous architecture styles using a notation.

The proposed structure adds the needed coherency to different evaluation methods. It can be used as a guideline for software architects in order to compare styles and even evaluation methods, because it uses a unique process for evaluation [5].

4.1. Evaluation Method

In this article, in order to obtain an evaluation about existing and heterogeneous methods, we make use of adaptation between quality attributes of styles and architecture users’ need, and exploit their quantity
attributes which have been obtained with respect to the background and functionality of each style since its creation. Combination of all these information gives more credit to our evaluation. In order to use these properties, we first describe the way of creating a tree of quality attributes. Then, for each architecture style, we construct a tree with respect to their effective quality attributes and architecture users' needs and obtain a comparative parameter for them. Finally we will discuss about the way of evaluating and describing heterogeneous architecture styles.

4.1.1. Constructing the tree of quality attributes

In order to construct a "quality attributes" tree we should consider quality attributes like performance, throughput, etc. as a parent node and put them in the root. Furthermore, we consider their effective parameters as their children. For example, in "Fig. 1" we put the quality attribute productivity in the root of tree and consider parameters like understandability, maintainability, and user friendliness as its children. $X_i$ is the effectiveness level of parameter $i$ (i.e. the level of being effective in the quality attribute).

![Fig. 1. Quality attribute tree](image)

In some cases, there are more detailed issues in the designer's mind and consider higher rank for them in comparison with other effective factors of a quality attribute. With respect to it, we consider another parameter $A_i$, which is defined as follows: if the user doesn't have any opinion in this level, $A_i$ is equal by one. Finally the operand $OP(Q_j)$ is defined by utilizing these information:

\[ OP(Q_j) = \sum A_i \cdot X_i \]

4.1.2. Constructing the tree of architecture styles

As mentioned before, some quality and quantity attributes have been listed for each architecture style. Each architecture style has a different coverage level of quality attributes with respect to its semantic model and structural specifications.

In "Fig. 2", $Q_j$ is a number between zero and one which has been obtained with respect to the background of each style. It indicates satisfaction level of the $j$th quality attribute in the style [18]. As we mentioned in section 4.1.1 Each quality attribute consists of a set of parameters; each parameter satisfies the quality attribute equal by the amount that $Q_j$ indicates.

![Fig. 2. Architecture style tree](image)

Another considered parameter is $B_i$. In constructing the quality attributes tree, we defined a parameter $A_i$ by which the architecture user ranked the impact factors of each quality attribute. A number is obtained for each architecture style by making use of these structures with use his method. It indicates the level of adaptation between current architecture styles and users' needs. The larger the obtained number is, the more the architecture style adapts with our needs.

\[ UP(A_i) = \sum UP(Q_j) \cdot Q_j \cdot B_i \]

An approach for more complicated domains (for establishing heterogeneous architecture)
Due to the complexity of problem scope of software architecture, we are forced to use more than one architecture styles together and simultaneously [16].

5. Classification of style combinations

Combining architecture styles in order to cover the domain problem is one of the important issues in architecture design. Generally we can classify these combinations into four categories: sequential, embedded, parallel, and hybrid.

5.1. Sequential heterogeneous styles

Putting architecture styles together in a way that when one part of system, which has a special style, finishes, another part with a different style will start. It is represented in "Fig. 3".

![Fig. 3. Sequential heterogeneous styles](image)

We define the operator "$\triangledown$" in order to represent the sequential relation of heterogeneous architecture styles as indicated below:

If $p$ and $q$ are architecture styles, then $p\triangledown q$ indicates sequential arrangement of them.
5.2. Embedded heterogeneous styles

Whenever a component of an architecture style has another style itself, we call that architecture style "embedded heterogeneous". In other words, the existence of an architecture style in another architecture style provides us with an embedded structure of architecture styles that we can use them to cover the needs and overcome complexities of problem domain. For example, in the layered architecture style illustrated in "Fig. 4", one layer has pipes-and-filters style.

![Fig. 4. Embedded heterogeneous styles(pipe & filter in layered)](image)

We define the operator "Δ" in order to represent the embedded relation of heterogeneous architecture styles as indicated below:

If p and q are architecture styles, then \( p \Delta q \) means that q is inside p.

5.3. Parallel heterogeneous styles

Whenever two or more architecture styles exist in a system structure without any interaction, we have parallel heterogeneous styles. It has been illustrated in "Fig. 5".

![Fig. 5. Parallel heterogeneous style](image)

We define the operator "\( p \parallel q \)" in order to represent the embedded relation of heterogeneous architecture styles as indicated below:

If p and q are architecture styles, then \( p \parallel q \) means that they are parallel with each other.

5.4. Hybrid heterogeneous styles

By virtue of the complexities of contemporary systems and their need for making use of styles that have enough coverage of the problem domain, sometimes we should use a combination of those three styles.

5.5. Example of notation

We imagine a system that includes three parts: preliminary, middle, and final. The first part follows pipeline architecture. The architecture of second part is layered; one of the layers follows the pipe-line architecture which has been imbedded in it. Finally, the outputs of these parts will enter to the final part which follows event-based style. In our notation this system is shown as below: (P: Pipe-line, L: Layered, E: Event-based)

\[ P \Delta (L \Delta P) \parallel E \]

6. Evaluating heterogeneous architecture of systems

Our goal is to evaluate the architecture of a system which has been composed of some different architecture styles. In order to represent the architecture of such systems by using our format (quality attributes tree construction and architecture styles tree), we should put the architecture of the system that we want to find a parameter to evaluate the heterogeneous architecture used in it, in the root of the tree. In the first level, we should put the architectural styles which are in the system sequentially or in parallel. Then, in the next levels, we put the styles which are embedded in each style of the previous level. It is illustrated in "Fig. 6". In this step, the architecture designer defines the parameter OD by virtue of the effectiveness level of each part of the system in the whole system; it determines the value of each part in the whole project.

![Fig. 6. Example of a tree's system heterogeneous architecture](image)

We can make use of the obtained tree structure to reach qualitative and quantitative evaluations by exploiting the methods which are suitable for our customized working domain. For example we can mention the use of genetic algorithm for searching architecture styles which have been stored by a tree structure in a repository Which will be implemented in future work. Generally, defining the notations and the obtained platform will lead to structured processes of software architecture design and evaluation. Moreover, it makes us able to perform their processes in a systematic framework.

7. Conclusion

In this article, after considering different approaches in software architecture evaluation and representing their strengths, weaknesses, and coverage level, we presented a framework and technique to evaluate architecture styles
that more than taking inherent quality attributes of each style into consideration, exploits the quantity attributes that have been obtained during the usage of them in order to increase the validity of the evaluation. Moreover, the architecture designer's viewpoint has been considered as an effective parameter in each step of validating effective factors in architecture styles in order to customize the process.

In the next step, we classified the current compositions and defined a notation to overcome the complexities of heterogeneous architectures and to become able to represent them explicitly. Furthermore, we become able to decompose them into smaller parts in order to use the defined operators and perform a suitable evaluation. This article, provides a general evaluation approach which covers the existing weaknesses by making use of inherent attributes of architecture styles in its evaluation process. It makes the system architect capable of composing different architecture styles clearly and simply and obtains a suitable evaluation of them. The constructed framework and platform could be used in some tools that are provided for architecture style selection such as [4] and generally can provide more precise and more applicable design and evaluation with respect to the requirements.

References
