Distilling Scenarios from Patterns for Software Architecture Evaluation – A Position Paper

Liming Zhu, Muhammad Ali Babar, Ross Jeffery

National ICT Australia Ltd. and University of New South Wales, Australia
{limingz, malibaba, rossj}@cse.unsw.edu.au

Abstract. Software architecture (SA) evaluation is a quality assurance technique that is increasingly attracting significant research and commercial interests. A number of SA evaluation methods have been developed. Most of these methods are scenario-based, which relies on the quality of the scenarios used for the evaluation. Most of the existing techniques for developing scenarios use stakeholders and requirements documents as main sources of collecting scenarios. Recently, architectures of large software systems are usually composed of patterns and styles. One of the purposes of using patterns is to develop systems with predictable quality attributes. Since patterns are documented in a format that requires the inclusion of problem, solution and quality consequences, we observed that scenarios are, though as informal text, pervasive in patterns description, which can be extracted and documented for the SA evaluation. Thus, we claim that the patterns can be another source of collecting quality attributes sensitive scenarios. This position paper presents arguments and examples to support our claim.

1 Introduction

The software architecture (SA) constrains the achievement of various quality attributes (such as performance, security, maintainability and modifiability) in a software intensive system [1, 2]. Since SA plays a crowning role in achieving system wide quality attributes, it is very important to evaluate a system's architecture with regard to desired quality requirements as early as possible. The principle objective of SA evaluation is to assess the potential of the chosen architecture to deliver a system capable of fulfilling required quality requirements and to identify potential risks [3].

A number of methods, such as Architecture Tradeoff Analysis Method (ATAM) [4] and Architecture-Level Maintainability Analysis (ALMA) [5], have been developed to evaluate a system's software architecture with respect to desired quality. Most of these methods are scenario-based. The accuracy of the results of these methods is largely dependent on the quality of the scenarios used for the evaluation [6]. The main sources of collecting scenarios are problem domain, quality requirements and stakeholders [1, 6]. We claim that architectural patterns and styles are another important source of collecting quality attributes specific scenarios.

Most of the software architectures for large and complex systems have embedded patterns. One of the major purposes of using patterns is to develop software systems that are expected to provide the desired level of quality [7]. Since patterns are

documented in a format that requires the inclusion of problem, solution, and quality consequences, we observed that patterns' description contain, though as informal text, scenarios and other architecturally significant information, which can systematically be extracted and appropriately documented to support the SA evaluation process.

In this paper, we present arguments why we believe that architectural patterns can be an important source for collecting scenarios and architectural related information. We also show how quality attribute sensitive general scenarios can be extracted from a few known architectural patterns. Our future research is aimed at formalizing the process of distilling scenarios from architectural patterns for architecture evaluation.

2. Motivation

SA evaluation and architectural patterns and styles are two sub-disciplines of software engineering, which have been gaining a lot of attention since early 90s [8, 9]. SA evaluation is important to predict the level at which the SA will support various quality attributes. Different techniques can be used for SA evaluation. Most of them are scenario-based as scenarios are very useful in characterizing quality attributes to be evaluated. For a scenario-based evaluation method, developing appropriate sets of scenarios are one of the most important activities [1].

The SA researchers have developed various techniques to develop scenarios that can be used to precisely specify and evaluate almost any quality attribute [4, 6, 10]. There are some inherent problems with these techniques; they are expensive, time consuming and the coverage of the final scenario sets is uncertain, which contributes to the possible sensitivity problem of evaluation methods [11]. That is why there is a need to find complimentary or alternative scenario collection techniques to support SA evaluation process.

Nowadays, the architectures of the large software systems are composed of patterns and styles [12]. Each pattern helps achieve one or more quality attribute in a system; however, each of them may also hinder other quality attributes. In pattern-oriented design, an architect develops a desirable SA by composing a number of architectural patterns and tactics. Patterns are documented in a format that requires the inclusion of problem, solution and quality consequences. That means within each pattern, there is information on the description of the scenarios that characterize the quality attributes being achieved by the pattern as well as the quality consequences of using the pattern.

These are the vital pieces of information required to perform SA evaluation and interim results of the evaluation. However, patterns are documented in a way that such information is not readily available to the software architect and SA evaluators. This may be the reason that the information within patterns is normally not used in SA evaluation. While there is a need to provide complimentary or alternative scenarios development techniques and there is huge amount of information implicitly hidden in pattern descriptions, we believe that distilling quality attribute specific information from the patterns can improve the SA evaluation process.

4. A Proposal

In the last section, we mentioned the major drivers of our research to find effective techniques to collect quality attribute specific general scenarios for SA evaluation and to utilize the architecture related information found in patterns. We believe one of the solutions to the afore-mentioned issue is to extract the architecturally important information from patterns and organize it into a format that it can readily be used during architecture design and evaluation. The availability of general scenarios for desired quality attributes during architecture design can help an architect to precisely articulate the quality requirements [7].

Most of the scenario-based SA evaluation methods require the stakeholders to generate scenarios to evaluate the architecture using requirement documents and brainstorming technique. We believe that if the stakeholders are provided with the general scenarios that characterize the quality attributes satisfied by the patterns used in the SA, it will improve SA evaluation and reduce the time and resources required to generate scenarios. Apart from general scenarios, there is another important piece of information which we call proto-evaluation. Proto-evaluations are the quality consequences for each quality attributes and tradeoffs made in the pattern. Proto-evaluations can be used for attribute analysis and tradeoff analysis.

5. Example

In this section, we show a few general scenarios extracted from known architectural patterns in EJB enterprise application [13]. We have stated earlier, a pattern has three elements: problem, solution and quality consequence. Scenarios are described mostly in problem element. However the quality attributes it concerns are also in quality consequence part since explicit quality attributes description are usually not elaborated extensively in the early part of the pattern especially the quality attributes bearing negative quality consequence. We have extracted the quality attribute sensitive scenarios using a scenario development framework proposed in [7]. This framework has following six elements:

- *Stimulus*
- *Response*
- *Source of the stimulus*
- *An environment*
- *A stimulated artifact*
- *A response measure*

For the details of each element, please see [7]. Stimulus, source of stimulus and environment can be found in the problem part of the investigated pattern. Response and stimulated artifact are commonly encountered in the solution part of the pattern. Explanations of the purpose of different parts within a pattern will reveal the stimulated artifact and expected response of the system. Response measures are usually pervasive, especially in the quality consequence part of the pattern documentation.

One scenario from Data Mapper pattern [13] is presented here:

*A periodic data structure change request (**stimulus**) from stakeholders (**source of the stimulus**) arrives when data use case changes after the system is deployed (**environment**). The system (**stimulated artifact**) has to be modifiable (**response**) according to the data structure change request within certain scope under certain time and maintenance cost (**response measure**).*

Similar general scenarios can also be extracted from Direct Access to Entity Bean, Data Transfer Object, Domain Data Transfer Object, Custom Data Transfer Object and Hash Factory [13]. However, all the extracted scenarios may not focus on the positive quality consequence. We can also extract scenarios by looking at negative quality consequence of a pattern and unexpected stimulus.

The second scenario has been extracted from the Data Transfer Object [13] pattern on data transfer performance:

*A periodic large amount of data requests (**stimulus**) from an independent source (**source of the stimulus**) arrive at the system under normal condition (**environment**). The system (**stimulated artifact**) has to transfer the data (**response**) within a certain amount of time under a certain network limit (**response measure**).*

Similar scenarios can be extracted from States Holder, Value Object, Detailed Object [13].

Both of the examples of scenario extraction from the architectural patterns are very high level general scenarios. Patterns usually have extra rich context sensitive information, which can be used to refine the general scenarios into more specific ones. For example, by integrating some contextual information, the performance general scenario can be refined to as following:

*A periodic large amount of requests on an individual data entity attribute (**stimulus**) from a user interface (**source of the stimulus**) arrive at the system under normal condition (**environment**). The system (**stimulated artifact**) has to transfer the data (**response**) within a certain amount of time without generating too many network calls (**response measure**).*

In order to make the general scenarios directly usable by SA evaluation, we need to convert them into concrete scenarios by providing system specific numbers for various elements like periodic, large, time and bandwidth etc.

6. Discussion and future work

This position paper argues that architectural patterns are an important source of collecting general scenarios and other architectural information to support the SA evaluation process. We have argued that there is valuable architecture related information, though as informal text, implicitly hidden in the patterns. This information can be systematically captured and used to improve the quality of the SA evaluation. This paper extracts and presents a quality attribute sensitive general scenario from known architectural patterns using a scenario development framework [7] to provide an example. Our future research is aimed at formalizing the scenario extraction process and providing a set of guidelines to identify, capture, and document general scenarios for SA evaluation.

7. References

[1] L. Bass, P. Clements, and R. Kazman, "Software Architecture in Practice," 2nd ed: Addison Wesley, 2003.

[2] P. Clements, R. Kazman, and M. Klein, *Evaluating software architectures : methods and case studies*. Boston: Addison-Wesley, 2002.

[3] N. Lassing, D. Rijsenbrij, and H. v. Vliet, "The goal of software architecture analysis: Confidence building or risk assessment," in *Proceedings of First BeNeLux conference on software architecture*, 1999.

[4] R. Kazman, M. Klein, G. Barbacci, and T. Longstaff, "The Architecture Tradeoff Analysis Method," 1998.

[5] N. Lassing, P. Bengtsson, J. Bosch, and H. V. Vliet, "Experience with ALMA: Architecture-Level Modifiability Analysis," in *Journal of Systems and Software*, vol. 61, 2002, pp. 47-57.

[6] P. Bengtsson and J. Bosch, "An Experiment on Creating Scenario Profiles for Software Change," University of Karlskrona/Ronneby, Sweden, 1999.

[7] L. Bass, F. Bachmann, and M. Klein, "Deriving Architectural Tactics-A Step toward Methodical Architectural Design," 2003.

[8] F. Bushmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, "Pattern-Oriented Software Architecture - A System of Patterns," John Wiley&Sons, 1996.

[9] M. Shaw and D. Garlan, "Software Architecture: Perspectivies on An Emerging Discipline," Prentice Hall, 1996.

[10] R. Kazman, G. Abowd, L. Bass, and P. Clements, "Scenario-based analysis of software architecture," in *IEEE Software*, vol. 13: Practical, 1996, pp. 47-55.

[11] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods," in *Software Engineering, IEEE Transactions on*, vol. 28, 2002, pp. 638-653.

[12] M. Fowler, "Patterns of enterprise application architecture," in *The Addison-Wesley signature series*. Boston: Addison-Wesley, 2003, pp. xxiv, 533 p.

[13] F. Marinescu, "EJB design patterns : advanced patterns, processes, and idioms." New York: John Wiley, 2002, pp. xxii, 259 p., [1] folded leaf.