Article can be accessed online at http://www.publishingindia.com

Software Architecture Evaluation: An Assessment

Shahkar Riyaz Tramboo*, S. M. K. Quadri**

Abstract

Software architecture is what defines a software system to be built. It starts early in the software development life cycle. The software architecture defines the data as well as the components of any software system along with the relation between them. It constitutes the blueprint that directs the development of the computer based software system. Being a critical activity of software development life cycle, any error in the design phase of software development can be critical to an organization dealing with the project and as such requires evaluation process that will not only analyse the architecture for its quality attributes but will benefit the software development organization by minimizing the risks associated with the software system to be built by pinpointing the errors early in the process of development. This paper highlights the architecture evaluation process with some examples of evaluation methods along with related work that has been previously done in the said field.

Keywords: Architecture Analysis, Architecture Trade-off Analysis Method, ARGUS-I, Empirically-based Software Architecture Evaluation, Layered Queuing Networks, Software Architecture Analysis Method, Software Evaluation Methods, Software Quality Attributes

Introduction

The foundation of any software is its architecture. The architecture of a computer based system is the first activity whereby the quality attributes such as security, reliability, usability, modifiability, stability, and real-time performance are addressed (Bahsoon & Emmerich, 2003). The software architecture defines the system as a well-defined structure having subcomponents and their corresponding interconnections. The design activity of architecture relating to computer software is one of the important phases of software development and it represents those decisions on design that are hard to alter. The design of architecture depends on a set of requirements that have been gathered during the requirement phase of software engineering (Mattson, Grahn & Martensson, 2006).

The structure of data and program components of a computer based system is defined by the architecture design activity of the software engineering. As defined by Bass (2007), the architecture of a software system is the configuration or formation, which involves element of a software system, the superficially evident qualities of those components, and the associations amongst them. The design action is the duty of the software designer who considers the appropriate style for the requirements specified as part of the software requirement and analysis phase.

The architectural design activity constitutes an important part of the software development life cycle; any decision made during this phase has a profound effect on the end product as it constitutes the blueprint over which the whole of the system is conceived. The software architectural design provides the big picture of the problem at hand and the various issues associated with it that needs to be addressed. The correct design selected among various alternatives during design phase leads to an end product

* Department of Computer Science, University of Kashmir, Jammu and Kashmir, India. E-mail shahkartramboo@gmail.com
** Department of Computer Science, University of Kashmir, Jammu and Kashmir, India.

which will satisfy required functions, quality attributes and will help the software designer to meet the set software reliability objectives. The bad design will not only lead to a product which will be fail to satisfy all requirements but also will cost a huge loss of time and money.

How to be sure whether the chosen architecture for any computer based system is correct? And whether it will not lead to a defective product or a product which satisfies the requirements defined as part of the requirement and analysis phase? As mentioned earlier the architecture is foundation for any computer based system. The design of software will permit or prohibit just about all of a system's functional and quality attributes. Modifiability, performance, security, availability, reliability objectives are set once the architecture is created. No amount of modification or smart application of trick will squeeze any of these qualities out of a defectively designed software product.

Software architecture evaluation methods provides a means to software architects to chose correct design and pave the way for the design of a accurate product. The software architecture evaluation methods are created to help the software architects to analyse various proposed architecture and select the appropriate one. The selection is generally based on the one that satisfies the various functional and quality attribute, setup during the analysis and design activity of the software architecture.

This paper showcases the need for software architecture evaluation with description of few of software architecture evaluation methods.

Related Work

Software design results in a number of architectural alternatives for the system to be created, that are each assessed to determine which is most appropriate for the system under design (Shaw & Garland, 1996). Shaw and Garland (1996) put forward that architecture is the first measure in software design, and scrutinizes the performance in which software architectural problems can affect software design and it also shows how to design new systems using architectural concepts already understood; highlights casual report, temporarily explaining official notations and terms, and the tools providing support; explains how to understand and have access to the design of existing system it from an architectural standpoint; and showcases existing example of system architectures that can be used as models to design new systems. Bahsoon and Emmerich (2003) stresses that there is a pressing need for evaluation of software architecture due to the volatile nature of software requirements and change of environment in which software works, and presents a value based reasoning approach for evaluation. Cook (2007) annotates that architecture evaluation helps an organization to judge whether the designed system satisfies the quality attributes notes during requirement and analysis phase, identifies design errors early in the software development life cycle (SDLC), and finds out probable technical threat in the project. Shaw and Clements (2006) in there survey examine the evolving trend of the software architecture research area and find that software architecture has entered into its golden era where it will be measured as an unremarkable and critical component of software system construction, otherwise taken casual, used with no ordeal, and supposed as a expected foundation for further progress.

Roy and Graham (2008) in their survey on different software architecture evaluation methods, review the necessity of varied evaluation techniques, connections and dissimilarity among them, their applicability, strong point and flaws. Clements, Klein and Kazman (2002) in their book Software Architecture Evaluation, showcase three different assessment methods all conceived at the institute of software engineering, which can be useful to software-demanding organizations, and demands the importance of assessment of architecture. Babar, Zhu and Jeffery (2004) document commonly known but informally explained qualities of evaluation methods and structured them within a planned framework that offered guidance on the choice of the most appropriate method for an evaluation exercise. They used this framework to characterise eight different evaluation methods. Griman, Perez, Mendoza and Losavio (2006) used analysis of feature of three unique evaluation methods of software architecture and the used it against a single case. The exercise ended up in two major offerings. First the features that needs to be present in an evaluation method of architecture and second the positive point of the evaluated methods. Li and Shou-Xin (2008) classify three types of software architecture evaluation techniques: scenario-based, metric and prediction based, and ADL-based methods. Characteristics of architecture evaluation method for software systems were then united inside these arrangements to create a framework for the purpose of comparison. The framework is then used

to examine a variety of existing software architecture evaluation methods pursued by stressing the issues that requires to be determined. Mattsson, Grahn, and Mårtensson (2006) examined several different evaluation methods for architecture assessment which deal with several different quality attributes and the evident trade-offs that lie between the different quality attributes. Breivold, Crnkovic and and Larsson (2012) presented a methodical re-evaluation of software architecture for software scalability. Their intention was to attain a general idea of the already evident techniques in analysing and enhancing software evolvability at architectural level, and explore the influence on research and exercise. Samarthyam, Suryanarayana, Sharma, and Gupta (2013) describe the motivation for MIDAS (Method for Intensive Design Assessments) as design assessment method at Siemens Corporate Development Center Asia Australia. From their experiences insight of MIDAS provided not only necessary pointers to different organizations and professionals in the design field to gain access and enhance the software design quality but also provided research queries to the software engineering society to discover. Harrison and Avgeriou (2013) in their report recommended pattern based architecture re-evaluation method for small-projects. They propose that in reviews of small projects, architecture patterns can be identified and used to identify potential problems related to satisfying the projects' important quality attributes. Their findings suggest that these reviews can be beneficial for small projects early in the development cycle that are unable or unwilling to undergo a comprehensive architecture review.

Software Architecture Evaluation

Software architecture presents the composition of data and component of a software product, the interrelationship between these components and the flow of data among them. Software architecture evaluation helps in analysing the properties of different software architecture styles produced during the designing phase of any software system. Evaluation process of architecture enables an architect to identify whether the various functional and quality attributes created are met in the design at hand, if not the alternative architecture created are considered for the selection process. Various architecture evaluation methods have been created for software architects so that they can choose and apply the one which will be beneficial for the creation of the product at hand.

Fig. 1: Software Architecture Evaluation Process

Quality attributes of any software system are not totally autonomous but may have positive or negative interaction between them e.g., performance of software can be affected by modifiability while on other side modifiability enhances reliability and security. The process of architectural evaluation helps to recognise, classify and analyse these trade-offs between various quality attributes (Mattsson, Håkan & Frans, 2006).

There exist various challenges in evaluation of software architecture. Like there exists a need for having an expert team for evaluation, which possesses the required experience in evaluation and documenting the report, also the documented design should follow a unified scheme which otherwise will lead to confusion among various stakeholder for understanding the design architecture for evaluation. Sometimes vaguely documented requirements for a software design also contribute to the evaluation problem as all the requirements are not communicated during requirement analysis which in turn leads to design without fulfilling quality requirements.

The methods of evaluating software architecture can be classified into four diverse groups namely experience-based, simulation-based, mathematical modeling based, and scenario based. The software architects can also use these different methods in combination in order to analyse different aspects of the design of software system (Reusner, Schmidt & Poernomo, 2003). The first group of methods relies on the experience and knowledge of the domain acquired previously by the designers (Mattsson *et al*., 2006). In case of simulation based methods, the implementation of the system is arranged, which in turn helps the designers to evaluate various quality and functional requirements for selection of the design at hand. The techniques in the set of mathematical modeling employ facts and technique that are mathematical in nature for evaluating runtime quality attributes such as performance and reliability of the components in the software architecture (Ionita, Hammer & Obbink, 2002). The last group of methods is based on creating a scenario

for verifying the particular attribute in architecture of a software system. Below are some of the evaluation methods:

Software Architecture Analysis Method (SAAM)

Software architecture analysis methods (SAAM) is an architecture evaluation method that falls in the group of scenario based evaluation methods (Vieira, Dias & Richardson, 2000). SAAM evaluates different quality attributes like maintainability, performance flexibility etc using scenarios and helps the designers to match them against the defined standards for comparison. SAAM helps to provide a quick look for these quality attributes in a software system. If a single architecture is to be evaluated it provides which quality attributes the architecture satisfies and which it does not. In case of evaluation of much different architecture it comes up with ranking of these architectures with respect to the fulfillment of their quality attribute requirements.

The primary inputs to SAAM are the various scenarios of how the user interacts with the system. The scenarios provide a means of description for specifying and evaluating qualities attributes. There exist six steps in the SAAM evaluation method. The first step constitutes the creation of creation of scenarios. Here all major elements like the users, uses, and all quality attributes are captured and the desired levels of each of these are noted with any future modification needed. Then follows the step where the description of architectures is done. Here the SAAM is fed with candidate architecture which describes the component, their interconnection and relation with the surroundings. Next the scenarios are classified and prioritised. The classification is done into direct and indirect scenarios. The prioritization is done based on voting which indicates the aspect that needs to be evaluated. Then the indirect scenarios are individually evaluated. Later the scenarios are accessed for interaction; if two scenarios interact over the same component; the component is modified in order to remove the interaction. At the last stage the different scenarios are weighted as per their success in fulfilling the system quality requirements.

Architecture Trade-off Analysis Method (ATAM)

Architecture trade-off analysis method (ATAM) is a scenario based evaluation method for software architecture for accessing the quality attributes like modifiability, flexibility maintainability etc. Unlike SAAM the ATAM analyses the interaction interdependency between quality attributes. This method helps to identify how much the quality attribute is satisfied and how much is the interdependency between the attributes. ATAM is based on SAAM. A set of prerequisites are needed for successful implementation of ATAM methods. First the architecture and its parameters need to be fully understood; trade-off points (critical for multiple attributes), sensitive points (critical for single attributes), and risks need to be identified and lastly quality attributes of the system need to be understood carefully.

ATAM constitutes four stages of implementation that are:

I. Presentation
II. Investigation and Analysis
III. Testing and,
IV. Reporting.

In the first step exchange of information between the stakeholders occurs through presentation regarding the system. In the subsequent phase the architects identify architectural approach, generate quality attribute utility tree, and analyse architectural approaches. In testing phase different scenarios are prioritised and architectural approaches are reanalysed. In the last phase, that is reporting phase, the results are presented to the stakeholders.

ARGUS-I

This is a specification based technique and is spotlighting both the component as well as architecture levels (Petriu, Shousha & Jalnapurkar, 2000). During architectural specification and implementation in case of ARGUS-I, iterative and scalable analysis are made possible. Both the structural and behavioral analyses are possible by blending synergistic combination of static and dynamic techniques. By static analysis authentication of architectural observance to design heuristics and incongruity between the data replaced between components and style policies are achieved, on the other hand defects in the dynamic component interaction and component to component communication may be revealed by dynamic analysis. ARGUS-I method of architecture analysis can be used to perform dependence analysis, interface mismatch, model checking, and simulation of architecture (Reusner *et al*., 2003).

The process stars by specifying the architecture and the components. The specified architectures are then evaluated for the set quality attributes. The performance in case of ARGUS-I simulation tool is determined based on the number of times the component or component interface is called upon during the simulation. The results are in the form of trace which can be presented and analysed graphically or in the text form.

Layered Queuing Networks (LQN)

LQN modal was the result of an extension development as part of well known queuing network model. It has been developed individually by several individuals (Lindvall, Tvedt & Costa, 2003). The LQN relies on the conversion of the architecture into what we call as layered queuing network model. The model illustrates the interactions between different architectural components and the time required for the required processing between them. The in-depth knowledge of various components of the architecture coupled with behavioural information is required. The more information regarding the system is fed to the model the more accurate the results are presented. The representative results of an LQN model include response times, throughput, utilisation of resources on behalf of different types of requests, and queuing delays. The results may be used to know the software and/or hardware blockage that maximizes the system performance under different loads and arrangements.

Empirically-based Software Architecture Evaluation (EBAE)

It pertains to an in-house development of a case study by Lindval *et al.* (2003) for redesigning the software system. The aim of doing so was to access the maintainability of the newly created system against the previously created versions. The author sketches out a process for empirically-based software architecture evaluation based on experience or observation alone and not on the theory. A number of metrics for architecture have been taken into account for the comparison and evaluation purposes.

The process involves the below mentioned steps:

- choose a viewpoint for the assessment
- label/choose metrics
- Pull together all the metrics,
- Assess and distinguish the architectures.

The evaluation standpoint of the case study was to calculate the maintainability, and the metrics that were used in the later study included organization, quantity, and union. The assessments were completed in a later phase of development that is when the implementation of the system was carried out.

Conclusion and Future Scope

The software systems architecture constitutes the foundation of any software system to be built upon, and thus requires a proper evaluation to be treated with. A proper evaluation will help the stake holders of the system to choose software architecture among the many submitted architectures, which will satisfy different functional quality attributes of a software system. Thus evaluation is an important activity that needs to be addressed with at most concern during the design phase of software development life cycle.

The software architecture evaluation methods that exist today deal with only one type of quality attribute for evaluation of any architecture and only a few exist that help us to check many different attributes at a time. Among the above mentioned methods there exists only one method that is ATAM that includes trade-off analysis between multiple attributes.

The evaluation methods need to be devised that will address more than one quality attributes and will help to perform tradeoffs and risks associated with the software system under construction. This will, in turn, help in recalibrating the design process for creation of fault tolerant systems.

References

Bahsoon, R., & Emmerich, W. (2003). *Evaluating software architectures: Development stability and evolution*. In Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia, (pp. 47-56.) IEEE Computer Society Press: Los Alamitos, US. Doi: 10.1109/AICCSA.2003.1227480.

Cook, D. (2007). *Architecture evaluation and review practices*. Retrieved from http://msdn.microsoft.com/en-us/library/bb896741.aspx

Bass, L. (2007). *Software architecture in practice*. Pearson Education India.

Mattson, M., Grahn, H., & Martensson, F. (2006). *Software Architecture Evaluation Methods for Performance, Maintainability, Testability, and Portability.* Retrieved from http://www.ide.bth.se/~hgr/Papers/qosa2006.pdf

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach* (6th ed.). New York, NY: McGraw-Hill Higher Education.

Roy, B., & Graham, T. C. N. (2008). Methods for Evaluating Software Architecture: A Survey. (Report No. 2008-545). School of Computing Queen's University at Kingston: Canada. Retrieved from http://techreports.cs.queensu.ca/files/2008-545.pdf

Shaw, M., & Garland, D. (1996). *Software architecture: Perspectives on an emerging discipline*. USA: Prentice Hall.

Shaw, M., & Clements, P. (2006). *The golden age of software architecture: A comprehensive survey*. Retrieved from http://reports-archive.adm.cs.cmu.edu/anon/isri2006/CMU-ISRI-06-101.pdf

Clements, P., Klein, M., & Kazman, R. (2002). *Evaluating software architectures: Methods and case studies.* USA: Addison-Wesley Longman Publishing.

Babar, M. A., Zhu, L., & Jeffery, R. (2004). *A framework for classifying and comparing software architecture evaluation*. In Proceedings of Australian Software Engineering Conference (ASWEC), (pp. 309-318). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.5504

Griman, A., Perez, M., Mendoza, L., & Losavio, F. (2006). Feature analysis for architectural evaluation methods. *Journal of Systems and Software*, 79(6), 871-881. Doi: http://dx.doi.org/10.1016/j.jss.2005.12.015

Li, Z. H. A. N. G., & Shou-Xin, G. H. W. (2008). Software architecture evaluation. *Journal of Software*.

Breivold, H. P., Crnkovic, I., & Larsson, M. (2012). A systematic review of software architecture evolution research. *Information and Software Technology*, 54(1), 16-40.

Samarthyam, G., Suryanarayana, G., Sharma, T., & S. Gupta. (2013). *MIDAS: A design quality assessment method for industrial software*. In Proceedings of the 2013 International Conference on Software Engineering, 2013, (pp. 911-920). Press Piscataway, NJ, USA.

N. B. Harrison, & P. Avgeriou. (2013). Using Pattern-Based Architecture Reviews to Detect Quality Attribute Issues-An Exploratory Study. In J. Noble, R. Johnson, U. Zdun, & E. Wallingford (Eds.), *Transactions on Pattern Languages of Programming III* (pp. 168-194). Berlin Heidelberg: Springer. Doi: 10.1007/978-3-642-38676-3_5

Roy, B., & Graham, T. C. N. (2008). Methods for evaluating software architecture: A survey. *School of Computing TR*, 545, 82.

Mattsson, M., Håkan, G., & Frans, M. (2006). *Software architecture evaluation methods for performance, maintainability, testability, and portability*. In 2nd International Conference on the Quality of Software Architectures.

Reusner, R., Schmidt, H. W., & Poernomo, I. H. (2003). Reliability prediction for component-based software architectures. *Journal of Systems and Software*, 66(3), 241-252.

Ionita, M. T., Hammer, D. K., & Obbink, H. (2002). *Scenario-Based Software Architecture Evaluation Methods: An Overview*. Department Software Architectures, Philips Research, Mathematics and Computing Science, Technical University 2002. Book chapter: Evaluating Software Architecture.

Vieira, M. E. R., Dias, M. S., & Richardson, D. J. (2000). *Analyzing software architectures with Argus-I.* In Proceedings of the 22nd International Conference on Software Engineering, (pp. 758-761).

Petriu, D., Shousha, C., & Jalnapurkar, A. (2000). *Architecture-based performance analysis applied to a telecommunication system*. IEEE Transactions on Software Engineering, 26(11), 1049-1065.

Lindvall, M., Tvedt, R. T., & Costa, P. (2003). An empirically based process for software architecture evaluation. *Empirical Software Engineering*, 8(1), 83-108.