Comparison of Scenario-Based Software Architecture Evaluation Methods

Muhammad Ali Babar, Ian Gorton
National ICT Australia Ltd. and University of New South Wales, Australia
{malibaba, ian.gorton}@nicta.com.au

Abstract

Software engineering community has proposed several methods to evaluate software architectures with respect to desired quality attributes such as maintainability, performance, and so on. There is, however, little effort on systematically comparing such methods to discover similarities and differences between existing approaches. In this paper, we compare four well known scenario-based SA evaluation methods using an evaluation framework. The framework considers each method from the point of view of method context, stakeholders, structure, and reliability. The comparison reveals that most of the studied methods are structurally similar but there are a number of differences among their activities and techniques. Therefore, some methods overlap, which guides us to identify five common activities that can form a generic process model.

1. Introduction

It has been shown that software architecture (SA) constrains the achievement of various quality attributes (such as performance and maintainability) in a system [1]. Several approaches have been proposed to address quality related issues at the SA level. Scenario-based approaches, a category of evaluation methods, are considered quite mature [2, 3]. There are also some attribute model-based methods and quantitative models for SA evaluation (for example, [4-6]), but, these methods are still being validated and are considered complementary techniques to scenario-based methods.

As existing methods are maturing or disappearing and new ones emerging, terminology, concepts, application domains, and activities are diverging. Therefore, it is becoming difficult to find out the differences and similarities among different methods. There is little work on systematically evaluating or comparing the existing methods and identifying a set of desirable features. We believe that a systematic comparison of SA evaluation methods can enhance the understanding of the methods' users and help researchers identify potential research directions.

The methods considered for this study include the Scenario-Based Architecture Analysis (SAAM) [7], the Architecture Level Modifiability Analysis (ALMA) [8], the Performance Assessment of Software Architecture (PASA) [9], and the Architecture Trade-off Analysis Method (ATAM) [10]. We mention the criteria used to include these methods and exclude others in section 2.

The purpose of this investigation is twofold: to extend our work on developing a method classification and comparison framework reported in [2] and describe the state-of-the-art in current scenario-based SA evaluation methods and future trends. We believe this work can help practitioners and researchers to understand and contrast alternative approaches that are available to them to evaluate a SA. We do not attempt to provide an exhaustive survey of SA approaches. Nor do we present this work as a method selection tool. However, we believe this work can provide some guidance on the choice of the most appropriate method for an evaluation exercise and opens up a basis for creation of a method selection instrument.

2. Background Work

Any attempt to present a comparison based on an overview of the state-or-the-art in a particular domain of research and practice usually starts from the findings of other researchers and practitioners. We have made every effort to find and examine all the survey work done on scenario-based SA evaluation methods during the last decade. Work reported in [11] provides detailed guidance on performing SA assessment but it addresses a different problem than the one tackled in this paper. To the best of our knowledge there are few attempts [2, 3] to provide a comprehensive treatment of topic. None of the other published survey or comparison of SA evaluation methods provides an explicit framework for comparing the methods. Rather, these surveys have been published to support the need for developing a new evaluation method, e.g. Bahsoon and Emmerich

Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04)
1530-1362/04 $ 20.00 IEEE

included an overview of the available SA assessment methods in their seminal work on ArchOptions [12].

Clements et al. wrote a chapter on method comparison in [13], however, they only compared three evaluation methods (SAAM, ATAM, and ARID [14]), all developed by the Software Engineering Institute (SEI). Moreover, their comparison framework does not include a number of important attributes that an evaluation method should have, for example, SA definition, tool support, and so forth.

We regard [2, 3] as two of the first rigorous attempts to provide a taxonomy of this growing area of research and practice. However, both are limited in their scope. For example, the authors of [3] do not provide any detailed explanation for the components of their comparison framework, nor do they explicitly describe the reasons for including those particular components in their framework. Moreover, there have been significant advances in SA evaluation research since their work was completed four years ago. For example, assessment methods for non-traditional quality attributes (usability, stability etc.) are being developed. Other evaluation methods (e.g. ATAM) have been published in books [1, 13].

[2] purports to present seminal work on developing and assessing a reliable framework to classify and compare SA evaluation methods. We have improved the comparison framework reported in [2] by making some adjustments to the framework based on its comparison with similar attempts reported in [15, 16]. However, this paper does not elaborate on the comparison framework.

We have also excluded a number of methods that appeared in our previous work as we believe those methods are not being activity used or developed. Moreover, we have included a recently developed method to evaluate SA performance [9]. We selected the studied methods based on their continuous development, which is evident from frequently appearing case studies reporting the results of using the methods included in this study.

3. A Comparison Framework

We compare SA evaluation methods using a comparison framework shown in table 1 as an analytical tool. This framework draws upon a number of sources to justify the selection and formation of its components and elements. The first is our earlier work on classifying SA evaluation methods [2].

This work advances our continuous efforts to design and assess a reliable tool that can provide some guidance in selecting an appropriate method. This extended version of our framework includes all the elements presented in previous work. We have introduced three more elements and arranged each element within four components of the framework.

Table 1. The components and attributes of the framework and the evaluation questions

Component	Elements	Brief explanation
Context	**SA definition**	Does the method explicitly consider a particular definition of SA?
	Specific goal	What is the particular goal of the methods?
	Quality attributes	How many and which quality attributes are covered by the method?
	Applicable stage	Which is the most appropriate development phase to apply the method?
	Input & output	What are the inputs required and outputs produced?
	Application domain	What is/are the application domain(s) the method is mostly applied?
Stakeholders	**Benefits**	What are the benefits of the method to the stakeholders?
	Involved Stakeholders	Which groups of stakeholders are required to participate in the evaluation?
	Process support	How much support is provided by the method to perform various activities?
	Socio-technical issues	How does method handle non-technical (e.g. social, organisational issues)?
	Required resources	How many man-days are required? What is the size of evaluation team?
Contents	**Method's activities**	What are the activities to be performed and in which order to achieve the goals?
	SA description	What form of SA description is recommended (e.g., formal, informal, particular ADL, views etc.)?
	Evaluation approaches	What types of evaluation approaches are used by the method?
	Tool support	Are there tools or experience repository to support the method and its artefacts?
Reliability	**Maturity of method**	What is the level of maturity (inception, development, refinement or dormant)?
	Method's validation	Has the method been validated? How has it been validated?

The second source for our modified framework is the NIMSAD (Normative Information Model-based System Analysis and Design) evaluation framework [15]. According to NIMSAD, there are four essential components for method evaluation: method context, method users, method content, and validation of method and its deliverables. We have modified two of the components' names and elements according to our domain. We believe that SA evaluation method not only considers the method users, it also takes into

account the benefits and needs of other classes of stakeholders, including sponsors of the evaluation exercise. Also, elements of the last component generally enhance the confidence of the method user in a method's capability; hence we call it reliability instead of validity.

We have also observed that most of the elements of our framework can also be mapped onto the elements of an evaluation framework suggested by NIMSAD and [17], which increases our confidence in the capability of our framework as a comparison tool. Other sources of the framework include [16, 18], which are applications of evaluation frameworks based on the work that forms the foundation of our work as well.
Our framework does not include an exhaustive list of questions that needs to be asked for method comparison. Rather, this framework can quite easily be enhanced, as is necessary in a nascent area.[19].

4. Overview of SA Evaluation Methods

4.1 Scenario-Based Architecture Analysis Method

The Software Architecture Analysis Method (SAAM) first time appeared in 1993 [7]. The goals of SAAM are mainly geared to evaluate SA against the desired quality attributes. SAAM can also compare different SAs with respect to given properties. SAAM was developed for modifiability [20] but it is being used for various quality attributes.

The most appropriate time to apply SAAM is after the high-level SA design and before implementation. Business drivers, SA description, and quality requirements are the main inputs to this method. The outputs of the method include quality sensitive scenarios, mappings between those scenarios and SA components, and the anticipated amount of effort associated with each change scenario. SAAM and its variants have been applied to in different domains, including CASE tools and combat systems [13].

The main benefits of SAAM are: early detection of problems, improved SA documentation and enhanced understanding of the SA issues. SAAM involves different stakeholders, e.g. architect, developer, maintainer and product manager. SAAM provides a number of techniques to perform various activities of the process, e.g. characterising quality attributes, eliciting scenarios, and classifying scenarios.

SAAM has six activities: scenario development, SA description, scenario classification and prioritization, individual scenario evaluation, scenario interaction, and overall evaluation. In the case of comparing multiple SAs, scenarios are assigned weightings to determine the overall ranking of different SAs. The first two activities are usually performed in parallel. SA description is captured using views proposed in [1].

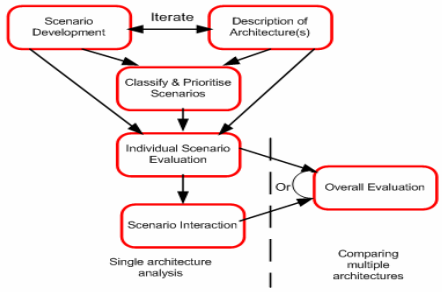

Figure 2. The process model of SAAM

SAAM evaluates each scenario by mapping it onto SA description and investigating whether the SA supports it (direct scenario) or not (indirect scenario). The cost of accommodating each indirect scenario is estimated by counting the number of required changes. Scenario interaction analysis reveals if many indirect scenarios affect the same component, a sign of poor separation of concern. SAAM is a mature approach, which has been validated with different case studies. Recently, SAAM has been superseded by ATAM [13].

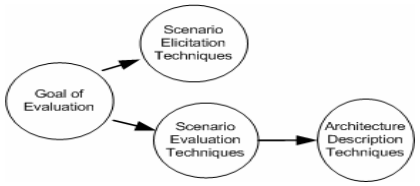

Figure 2. Goal-oriented evaluation concept of ALMA

4.2 Architecture Level Modifiability Analysis

The work of Bengtsson and Lassing on modifiability of SA resulted in Architecture Level Modifiability Analysis (ALMA) [21, 22].

ALMA has been developed around a conceptual framework that we call goal-oriented evaluation. Goal setting is the most important activity of this method as the rest of activities are performed in the light of the evaluation goals. Figure 2 shows the goal-based philosophy of the ALMA. The specific goal of this method is to address modifiability related issues at the SA level. The goals of modifiability can be:

- Maintenance cost prediction – estimating the effort required to satisfy software change scenarios
- Risk assessment – identifying the types of changes for which a SA is inflexible

- SA selection – comparing two or more candidate SAs to choose the better candidate.

ALMA is usually utilized before implementing the SA but there is no reason to assume that it is not suitable for legacy system reengineering projects. The inputs include SA specifications and quality requirements [23]. ALMA has successfully been applied in telecommunications, information systems, embedded systems and medical domains [24]. The main benefits of using ALMA are identification of SA risks, estimation of the efforts required to accommodate change, or selection of an optimal SA.

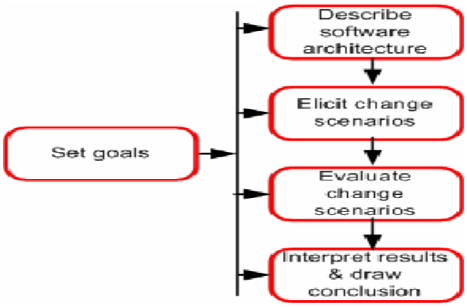

Figure 3. The process model of ALMA

ALMA usually involves only a small set of stakeholders, namely the development team and software architect. The method can be applied both top-down, starting from a predefined scenario classification, and bottom-up, starting from concrete scenarios and building up categories of scenarios. ALMA provides techniques to select relevant scenarios and to reduce the number of scenarios [24]. It also provides guidance on when to stop generating scenarios. ALMA consists of setting goals, describing the SA, eliciting scenarios, evaluating scenarios, and interpreting results and drawing conclusions (Fig 3 shows method's activities). The method uses UML along with various SA views to describe a SA [25].

ALMA uses impact analysis to evaluate the SA against change scenarios. Impact analysis is performed by identifying the components affected by the scenarios, figuring out the required modifications, and determining ripple effects. The results are interpreted depending on the goal of evaluation. ALMA provides a framework to describe results quantitatively. As ALMA has been validated with several applications, the method is considered quite mature.

4.3 Performance Assessment of Software Architecture

Williams and Smith presented a method to assess performance related issues at SA level in [9], called Performance Assessment of Software Architecture (PASA). This method has been proposed based on their work on techniques and tools for performance evaluation of SA reported in [26, 27]. PASA includes performance sensitive SA styles and anti-patterns as analysis tools and formalizes the SA analysis activity of the performance engineering process reported in [28]. Another major difference between Williams and Smith's earlier work on performance assessment of SA and this work is additional focus on client interaction and information gathering strategies. [29].

The specific goal of PASA is to assess the capability of candidate SA(s) with respect to performance objectives of a system. PASA guides the SA analysis activity using performance related scenarios as source of reasoning. Additionally, the analysis also considers other quality attributes (e.g. maintainability) as well and trade-offs that need to be made [9]. PASA has also been used to compare different SAs [26].

PASA can be applied early in the development cycle, post-deployment, or during an upgrade of a legacy system. The method has been applied to Web-based systems, embedded systems, real-time systems, and in the financial domain [29]. PASA needs SA descriptions documented using various views [30]. If the SA is not well-documented, a common problem [13], architectural information is extracted from developers, software code, and other artefacts. Only the development team is usually involved.

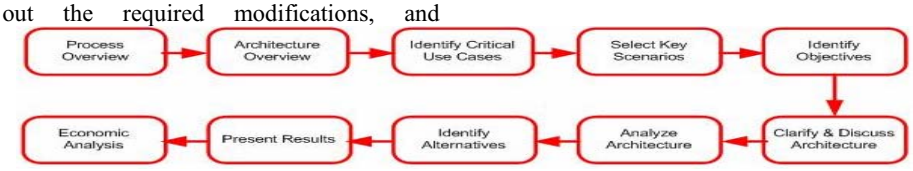

Figure 4. The process model of PASA

PASA has ten steps shown in Fig. 4. The evaluation starts with a process presentation session aimed at setting the goals, identifying the information required, finding stakeholders' expectations, and describing the

various aspects of the method. During the next step, the evaluators get a high level overview of the SA without any details. If a SA is not well-documented, it is also documented.

The next step tries to identify critical use cases. From a performance evaluation perspective, critical use cases are those for which there is significant performance risk. The evaluation team work with the developers to select key performance scenarios within each use case. Following the general practices of SPE, PASA requires the selected scenarios to be documented using augmented UML sequence diagrams [31]. Each key scenario usually has one or more goals associated with it. Performance objectives can be described in terms of response time, throughput, or constraints on resource usage. The SA discussion provides another opportunity to gain further information on the SA. The evaluation team may also collect performance measurement data and metrics [9].

The next step is aimed at identifying architectural styles or patterns used in the SA. If there is any deviation from the archetype of the style or pattern, the evaluators try to determine if there is any negative effect caused by that deviation. If there are any *antipatterns* [32] found, the evaluators perform *refactoring*. PASA also uses different quantitative techniques for performance modelling including software and system execution models. The process finishes with a presentation of the results to the clients and economic analysis of the assessment exercise. The later activity is important to justify the cost and highlight the benefits [29].

This method incorporates both qualitative and quantitative techniques to illustrate the potential risks that may be inherent in a SA. This method also demonstrates how scenarios can be useful in characterising run-time quality attributes like performance. PASA itself or its various techniques have been validated with different case studies [28].

4.4 Architecture Trade-off Analysis Method

The Architecture Trade-off Analysis Method (ATAM) was initially positioned as a SA design method [10] to support design trade-offs. Later, it was presented as a model for SA analysis.

The specific goal of ATAM is to promote disciplined reasoning for analysing a SA's capability with respect to multiple quality attributes. It also helps make trade-offs between competing attributes. ATAM claims to be applicable during any stage of the software development, however, it is most effective when applied to the final version of a SA. The inputs for ATAM include business goals, software specifications, and SA description. The outputs of ATAM are list of scenarios, sensitivity points, trade-off points, risks, SA approaches, and so on.

The application domains include combat systems, web-based systems and embedded systems. ATAM claims to provide several technical as well as social benefits. ATAM involves various stakeholders.

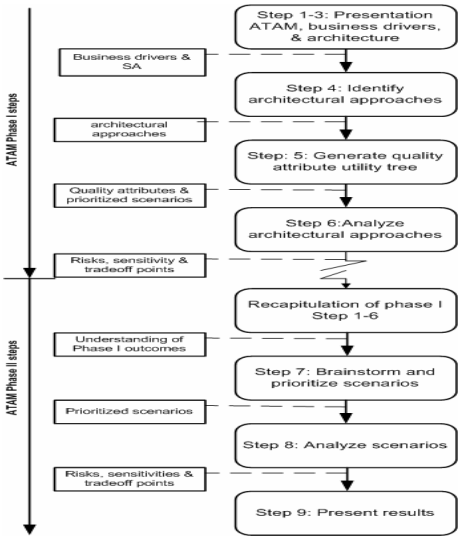

Figure 5. The process model of ATAM

ATAM is a heavy weight process that consists of four phases. There are nine activities in those phases (Fig 5). There are a number of activities, which are repeated in phase I and II. First these activities only involve selected stakeholders, usually technical staff of the project. During the second phase a wide range of stakeholders are invited. ATAM requires a SA documented with different views [1].

ATAM does not prescribe any specific evaluation techniques. Rather, it uses various theoretical models of the quality attribute communities for quantitative analysis and applies qualitative reasoning heuristics documented in terms of attribute-based SA styles (ABAS) [5], architectural patterns, tactics or quality sensitive scenarios [1]. ATAM is considered a mature approach as it has been validated in different domains. A tool support, ArchE, is underdevelopment [33].

5. Method Comparison

5.1 Context

A precise and well-documented definition of a SA is very important for a successful SA evaluation [34]. It is difficult to define metrics to assess the capability

of a SA with respect to quality attributes without precisely describing the SA according to a particular evaluation method [35]. All of the methods leave a SA undefined under the assumption that everyone knows what SA means.

There is at least one common goal found in all the methods, which is prediction-based assessment of the quality of a system at the SA level. However, each has a specific view and different approach to achieve the goal: SAAM is mainly geared to identify the potential SA risks; ALMA specializes in predicting one quality attribute i.e., modifiability and there are three possible objectives to be pursued: risk assessment, maintenance cost prediction, and SA comparison; PASA studies a SA to identify and mitigate performance related risks. ATAM identifies and analyses sensitivity and trade-off points as these can prevent the achievement of a desired quality attribute.

One of the most significant features of method comparison is the number of quality attributes a method deals with. Most of the scenario-based methods focus mainly on a single quality attribute. ALMA is aimed at modifiability analysis, PASA focuses on SA performance analysis, while SAAM was developed to assess modifiability. Amongst the studied methods, ATAM is the only method that considers multiple quality attributes. ATAM focuses on those decisions that affect (positively or negatively) one or more quality attributes, which are called either sensitivity or trade-off points depending upon the number of attributes affected by a decision.

SA evaluation is traditionally performed after the specification of the SA and before the beginning of the implementation. This common practice is evident from the comparison of the methods as well. From this perspective, all of the compared methods are applied to the final version of the SA. ATAM is also used as a SA design and analysis method in architecture-based development. However, most of these methods claim to be equally applicable to any other stage of the development lifecycle.

SAAM, ALMA and ATAM share a number of inputs and outputs, including requirements specifications, business drivers and SA descriptions. PASA needs similar inputs but in different form, there are a number of common outputs among the studied methods, such as scenarios, SA approaches, risk-spots and so on. However, ATAM produces a number of other artefacts, namely sensitivity points, trade-off points and utility trees.

There are several different domains in which these methods are being applied. Embedded systems, telecommunications, and information systems seem common domains among the surveyed method. However, SAAM and ATAM differentiate themselves based on their use for combat and avionics systems.

5.2 Stakeholders

A stakeholder is any person or organisational representative who has a vested interest in a system [36]. The studied methods also vary in terms of number and categories of stakeholders involved in evaluation. For example, SAAM and ATAM involve all major stakeholders, including architects, designers, and end users, while ALMA usually depends on the architecture designer and rarely involves other stakeholders. PASA focuses on the developers. It may involve maintainers as well.

All of the studied methods provide at least a coarse-grained description of the evaluation process. However, detailed guidance is sparse. Only ATAM provides sufficient process instructions. Other methods describe the required activities, however, do not elaborate on the suitable techniques for each activity.

SA evaluations are greatly influenced by non-technical issues like organisational structure, communication channels, stakeholders' vested interests, political factors, and managerial concerns. Only ATAM stands out from its counterparts in terms of its detailed guidelines and techniques to deal with social issues. Some methods briefly mention social issues without suitably dealing with them.

Most of the surveyed methods do not provide any explicit information on the cost of an evaluation or the resources required. Two methods (SAAM, and ATAM) mention the desirable shape of the evaluation team and various stakeholders, however, there is hardly any information about other resources or the cost of using these methods.

5.3 Contents

In scenario-based methods, there are a number of activities that appear to be the same at the coarse-grained level; however, a fine-grained analysis of those activities reveals a number of differences. For example, scenario development and scenario evaluation activities are common in scenario-based methods, but the techniques of performing these activities are quite different. For example, ALMA uses scenario profiles to categorise the generated scenarios; ATAM provides a six element framework to characterise quality attributes, and uses a utility tree for generating and classifying scenarios; PASA uses both use cases and scenarios to identify performance goals.

Communicating a SA to its stakeholders is one of the critical factors of a successful SA evaluation

exercise. Different Architectural Description Languages (ADLs) have been developed [19], and a SA is also documented using various views [1, 37, 38]. None of the studied methods prescribes any particular ADL; all of them use SA views, however, the number and type of views vary from method to method. For example, logical and module views may suffice for SAAM, but process, data-flow, user, physical, module and many more may be required by the ATAM.

The studied methods can be compared based on their fine-grained techniques. SAAM is purely scenario-based, ALMA uses a variety of approaches depending on evaluation goals, PASA combines scenarios with performance modelling, and ATAM applies attribute model-based analysis. ALMA also provides analytical models for modifiability, while others use those provided in [39, 40].

There is a need for automating as many tasks of SA design and evaluation as possible [2]. A tool can also capture the design artefacts along with the decision rationale, evaluation outcomes, measurement and administrative information that are invaluable assets.

All the studied methods recognise the importance of appropriate tool support, however, only SAAM provides a tool (SAAMTOOL) [41] to partially support the evaluation process. There will be a tool available for ATAM soon [33]. Another aspect of automation is knowledge management for reusability, which is recognised as one of the most important means of increasing productivity, quality and cost-effectiveness [42]. Only ATAM provides guidance on generating and utilizing the reusable artifacts, i.e., identified risks, scenarios, quality attributes etc. It also recommends a repository of the artifacts [13].

5.4 Reliability

SA evaluation methods can also be compared from the point of its maturity as it may foster confidence in method users. We believe that existing evaluation methods can be classified in any of the four maturity phases of SA evaluation methods lifecycle, namely inception, development refinement and dormant [2]. ATAM and ALMA can be considered in the refinement stage. SAAM and PASA can be considered in development stage.

The process of method development and the techniques used to validate it may encourage or discourage the evaluators to select one particular method over the other[43]. All of the methods have been validated in several domains.

4. Conclusions

The main contribution of this paper is systematically studying four scenario-based SA evaluation methods using an extended version of a comparison framework [2]. We have also demonstrated that the framework is modifiable by extending it based on a simple comparison with similar work in other domains [15, 16].

The comparison reveals several features supported by most of the methods. An example is suitable guidance on required SA description and views. Most of them also provide appropriate techniques for quality attribute characterisation and scenario generation and evaluation. The survey also highlighted a number of issues which existing methods do not sufficiently address. Only one method, ATAM, provides comprehensive process support. Social aspects of the evaluation are given sparse attention. No working definition of the SA is explicitly provided. Finally, tool support for the evaluation process is almost non-existent. Furthermore, the comparison also revealed that some methods overlap, which guides us to identify five common activities that can form a generic process for SA evaluation. The common activities are:

1. Evaluation planning and preparation.
2. Explain SA approaches.
3. Elicit quality sensitive scenarios.
4. Analyze SA approaches.
5. Interpret and present results.

5. References

[1] Bass, L., et al., "Software Architecture in Practice", 2 ed. 2003: Addison-Wesley.

[2] Ali-Babar, M., et al., "A Framework for Classifying and Comparing Software Architecture Evaluation Methods," *Proc. of the Australian Software Engineering Conference, Melbourne, Australia*. 2004.

[3] Dobrica, L. and E. Niemela, "A Survey on Software Architecture Analysis Methods," *IEEE Transactions on Software Engineering*, 2002. **28**(7).

[4] Svahnberg, M., et al., "A Method for Understanding Quality Attributes in Software Architecture Structures," *Proc. of the 14th Int.l Conf. on Software Eng. and Knowledge Eng.* 2002. Ischia, Italy.

[5] Klein, M. and R. Kazman, "Attribute-Based Architectural Styles," Tech. Report CMU/SEI-99-TR-022, Soft Engineering Institute, Carnegie Mellon University, 1999

[6] Duenas, J.C., et al., "A Software Architecture Evaluation Model," *2nd Int.l Workshop On Development and Evolution of Software Architectures for Product Families*. 1998.

[7] Kazman, R., et al., "SAAM: A Method for Analyzing the Properties of Software Architectures," *Proc. of the 16th ICSE*. 1994.

[8] Bengtsson, P., et al., "Architecture-level modifiability analysis (ALMA)," *Journal of Systems and Software*, 2004. **69**(1-2).
[9] Williams, L.G. and C.U. Smith, "PASA: A Method for the Performance Assessment of Software Architecture," *Proc. of the 3rd Workshop on Software Performance*. 2002. Rome, Italy.
[10] Kazman, R., et al., "The Architecture Tradeoff Analysis Method," *Proceedings of IEEE, ICECCS*. 1998.
[11] Kruchten, P., "Software Architecture Review and Assessment (SARA) Report," *Proc. of the Software Architecture Review and Assessment Workshop*. 2002. Orlando, Florida.
[12] Bahsoon, R. and W. Emmerich, "Evaluating Software Architectures: Development, Stability, and Evolution," *Proceedings of ACS/IEEE Int. Conf. on Computer Systems and Applications*. July, 2003. Tunis, Tunisia.
[13] Clements, P., et al., "Evaluating Software Architectures: Methods and Case Studies". 2002: Addison-Wesley.
[14] Clements, P.C., "Active Reviews for Intermediate Designs," Tech. Report CMU/SEI-2000-TN-009, SEI, Carnegie Mellon University, 2000
[15] jayaratna, N., "Understanding and evaluating methodologies: NIMSAD: a systematic framework". 1994, London: McGraw-Hill.
[16] Forsell, M., et al., "Evaluation of Component-Based Software Development Methodologies," *Proc. of FUSST*. 1999. Tallinn.
[17] Kronlof, k., "Method Integration: Concepts and Case Studies". 1993: John Wiley & Sons.
[18] Matinlassi, M., "Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, KobrA and QADA," *Proc, of the 26th Int'l Conf. Software Eng.* 2004. Edinburgh, Scotland.
[19] Medvidovic, N. and R.N. Taylor, "A Classification and Comparison Framework for Software Architecture Description Languages," *IEEE Transactions on Software Engineering*, Jan, 2000. **26**(1): p. 70-93.
[20] Kazman, R., et al., "Analyzing the Properties of User Interface Software," Tech. Report CMU-CS-93-201, School of Computer Science, Carnegie Mellon University, 1993
[21] Lassing, N., et al., "Towards a Broader View on Software Architecture Analysis of Flexibility," *Proceedings of 6th Asian-Pacific Software Engineering Conference*. 1999.
[22] Bengtsson, P. and J. Bosch, "Architectural Level Prediction of Software Maintenance," *Proceedings of 3rd European Conference on Software Engineering Maintenance and Reengineering*. 1999.
[23] Lassing, N., et al., "How Well can we Predict Changes at Architecture Design Time?," *Journal of Systems and Software*, 2003. **65**(2).
[24] Bengtsson, P., "Architecture-Level Modifiability Analysis," Ph.D. Thesis, Blekinge Institute of Technology, Sweden, 2002
[25] Lassing, N., et al., "Using UML in Architecture-Level Modifiability Analysis," *Proceedings of the Workshop on Describing Software Architecture with UML, ICSE*. 2001.
[26] Williams, L.G. and C.U. Smith, "Performance Evaluation of Software Architectures," *proc. of the Workshop on Software and Performance*. 1998. Santa Fe, USA.
[27] Smith, C.U., "Performance Engineering for Software Architectures," *Proc. of the 21th Computer and Software Applications*. 1997. Washington, DC, USA.
[28] Smith, C.U. and L.G. Williams, "Performance Solutions: A Practical Guide to Creating Responsive, Saclable Software". 2002: Addison-Wesley.
[29] Williams, L.G. and C.U. Smith, "PASA: An Architectural Approach to Fixing Software Performance Problems," *Proc. of Int. Conference of the Computer Measurement Group*. 2002. Reno, USA.
[30] Kruchten, P.B., "The 4+1 View Model of architecture," *Software, IEEE*, 1995. **12**(6): p. 42-50.
[31] Booch, G., et al., "The Unified Modeling Language User Guide". 1999: Addison-Wesley.
[32] Brown, W.J., et al., "AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis". 1998: John Wiley.
[33] Bachmann, F., et al., "Preliminary Design of ArchE: A Software Architecture Design Assistant," Tech. Report CMU/SEI-2003-TR-021, SEI, Carnegie Mellon University, 2003
[34] Bot, S., et al., "A Stakeholder-Centric Software Architecture Analysis Approach," *Proceedings of 2nd International Software Architecture Workshop*. 1996.
[35] Avritzer, A. and E.J. Weyuker, "Investigating Metrics for Architectural Assessment," *Proc. of the 5th Int'l Software Metrics Symposium*. 1998. Bethesda, Maryland.
[36] Kruchten, P., "Rational Unified Process: An Introduction". 2000: Addison-Wesley.
[37] Hofmeister, C., et al., "Applied Software Architecture". 2000, Reading, MA: Adison-Wesley Longman.
[38] Kruchten, P., "The 4+1 View Model of Architecture," *IEEE Software*, Nov. 1995. **12**(6).
[39] Lyu, M.R., ed. *Handbook of Software Reliability Engineering*. 1996, McGraw-Hill and IEEE Computer Society: New York.
[40] Smith, C.U. and L.G. Williams, "Software Performance Engineering: A Case Study Including Performance Comparison with Design Alternatives," *IEEE Transactions on Software Engineering*, 1993. **19**(7).
[41] Kazman, R., "Tool Support for Architecture Analysis and Design," *Proc. of the 2nd Int'l Software Architecture Workshop*. 1996. Carlifornia, USA.
[42] Aurum, A., et al., eds. *Managing Software Engineering Knowledge*. 2003, Springer- Verlag: Berlin Heidelberg.
[43] Shaw, M., "The Coming-of_Age of Software Architecture Research," *IEEE - Computer*, 2001: p. 657-664.