

Assessment of a Framework for Comparing Software Architecture Analysis Methods

Muhammad Ali Babar¹, Barbara Kitchenham²

¹Lero, University of Limerick, Ireland, ²National ICT, Australia
Muhammad.AliBabar@ul.ie, Barbara.kitchenham@nicta.com.au

We have developed a framework, FOCSAAM, for comparing software architecture analysis methods. FOCSAAM can help architects and managers to choose a specific method to support architecture analysis process. We have been assessing the suitability of the framework's elements in different ways. During the development of FOCSAAM, a theoretical assessment was performed by relating each of its elements to the published literature on quality assurance, process improvement, and software development approaches. Moreover, we have also found that most of the elements of FOCSAAM can also be mapped onto the elements of a well-known framework for comparing information systems development methods, NIMSAD framework. Our goal of this study was to further assess the suitability of different elements of FOCSAAM by using the expert opinion approach. We asked 17 practicing architects with extensive experience to assess the suitability of the elements of FOCSAAM for selecting a particular method to support the software architecture analysis process. The findings of this study provide support for each element of FOCSAAM to be included in forming criteria for comparing software architecture analysis methods.

Keywords: Software architecture analysis, comparing methods, expert opinion, model assessment, empirical studies

1. INTRODUCTION

It has been shown that Software Architecture (SA) constrains the achievement of various quality attributes (such as performance, security, maintainability and usability) in a system (Bass et al., 2003). Since software architecture plays a significant role in achieving system wide quality attributes, it is important to analyse a system's software architecture with regard to desired quality requirements as early as possible (Dobrica and Niemela, 2002). The principle objective of software architecture analysis is to assess the potential of a proposed architecture to deliver a system capable of fulfilling required quality requirements and to identify any potential risks (Lassing et al., 1999). Additionally, it is quicker and less expensive to detect and fix design errors during the initial stages of the software development. That is why an effective method to analyze prospective software architectures is of great business value (Abowd et al., 1997).

Several methods have been developed to support the software architecture analysis process (Ali-Babar et al., 2004, Ali-Babar and Gorton, 2004, Dobrica and Niemela, 2002). These methods have several commonalities regarding the artifacts, activities, and stakeholders involved in the assessment process. However, since it is not clear which of these methods is most effective in achieving their shared goals, an objective mechanism to analyse these methods is required. To address this issue, we have identified a set of features that can provide a basis for assessing and comparing software architecture analysis methods and organised them within a framework (Ali-Babar and Gorton, 2004). The Framework for Comparing Software Architecture Analysis Methods (FOCSAAM). FOCSAAM consists of seventeen elements, which have been organised into four components. FOCSAAM provides a basis for systematically assessing strengths and weaknesses of available software architecture analysis methods. This framework is also expected to offer guidance on the selection of an appropriate method for analysing software architecture in a particular context.

We have performed a theoretical assessment of each of the elements included in FOCSAAM. The theoretical assessment of the framework was performed by relating each of its elements to the published literature on quality assurance, process improvement, and software development approaches (Ali-Babar et al., 2004). Moreover, we have also observed that most of the elements of FOCSAAM can also be mapped onto the elements of a well-known framework for comparing information systems development methods, the NIMSAD framework (Jayaratna, 1986), which increases our confidence in the capability of FOCSAAM as an effective tool for software comparing architecture analysis methods. This paper reports the design and outcomes of an empirical study aimed at further assessing the suitability of different elements of FOCSAAM by using an expert opinion approach, which asks practicing architects to assess the suitability of the elements of FOCSAAM as checklist questions to be asked while selecting a particular method to support the software architecture analysis process.

2. BACKGROUND AND MOTIVATION

Software architecture evaluation has emerged as an important quality assurance technique. Being a new research area, the number of methods proposed to analyse software architecture with respect to desired quality attributes is continuously increasing. As mentioned earlier, there is a need for a mechanism to provide systematic guidance on classifying and comparing different methods based on their features. Therefore, we have been developing and refining a method comparison framework by studying the current scenario-based software architecture evaluation methods. We have attempted to cover as many points of view as possible with the sole purpose of identifying the similarities and differences among these methods. We have developed a set of questions and put them into a framework, FOCSAAM, which can be used in an effective manner by an architect or project manager to choose a specific method for the software architecture evaluation process.

2.1 Related Work

Any attempt to provide a taxonomic comparison based on a comprehensive overview of the state-of-the-art in a particular area of research and practice is normally based on discoveries and conclusions of other researchers and practitioners and other previous surveys. We regard the work reported in (Dobrica and Niemela, 2002) as the first comprehensive attempt to provide a taxonomy of this growing area of research and practice. However, their work has its own shortcomings. For example, the authors 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.

Kazman et al. have also provided criteria for analyzing software architecture evaluation methods (Kazman et al., 2005). Their criteria are intended to help analyze the effectiveness and usability of architecture analysis methods. These four criteria are: context and goal identification; focus and properties under examination; analysis support; and analysis outcomes. Before proposing that criteria Kazman et al., critically examined FOCSAAM. Though, they agree that FOCSAAM provides a vital catalogue of assessment criteria for any organisation looking to adopt an analysis method, they argued that their criteria is aimed at much more deeper comparison (Kazman et al., 2005). However, we contend that a detailed analysis of their criteria reveals that most of the elements of FOCSAAM are underpinning that four point criteria of analysing architecture analysis methods. Moreover, we do not claim that we have produced an exhaustive list of features that can be used to assess different architecture analysis methods. Rather, we have identified key elements that could be used as a catalogue of assessment criteria for comparing architecture analysis methods. The following section briefly describes the development and assessment process of FOCSAAM.

TABLE 1. THE COMPONENTS AND ELEMENTS 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?
Stakeholders	Application domain	What is/are the application domain(s) the method is mostly applied?
	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)?
Contents	Required resources	How many man-days are required? What is the size of evaluation team?
	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 required (e.g., formal, informal, ADL, views etc.)?
	Evaluation approaches	What types of evaluation approaches are used by the method?
Reliability	Tool support	Are there tools or experience repository to support the method and its artefacts?
	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?

2.2 FOCSAAM Development and Assessment

FOCSAAM is shown in Table 1. We identified the components and elements of FOCSAAM by extensively reviewing the literature on software engineering methods, processes in general, and software architecture evaluation methods in particular (Ali-Babar et al., 2004). Some of the components of the framework were also identified by discovering commonalities and differences among the existing architecture evaluation methods. We have also drawn upon a number of other sources including literature reviews of the architecture evaluation methods (Dobrica and Niemela, 2002, Clements et al., 2002), and heuristics of experienced software architects and software engineers. We also based the structural form of FOCSAAM on a comparison framework for information system development methods, NIMSAD (Normative Information Model-based System Analysis and

Design) (Jayaratna, 1986). Detailed discussions on the selection, definition, and justification of each of the elements of FOCSAAM have been reported in (Ali-Babar et al., 2004, Ali-Babar and Gorton, 2004).

We have also observed that most of the elements of our framework are semantically equivalent to the fundamental elements included in the NIMSAD framework (Jayaratna, 1986), which increases our confidence in the capability of our framework as a comparison tool. However, unlike NIMSAD, FOCSAAM is focused on software architecture evaluation process and provides those elements which characterise software architecture evaluation methods. We also performed theoretical evaluation of the framework by relating each of its elements to the published literature on quality assurance and process improvement approaches (Ali-Babar et al., 2004). Moreover, two other sources that provide the justification for different components and elements of the framework are (Matinlassi, 2004, Forsell et al., 1999). These are applications of evaluation frameworks based on the work (Jayaratna, 1986) that forms the foundation of our work as well.

3. EXPERT OPINION BASED ASSESSMENT

This section presents the design and results of a qualitative study that used the expert-opinion technique to assess different components of FOCSAAM. Expert opinion (also called expert judgment) is a systematic approach to obtaining information or answers to specific questions about certain quantities, called issues, such as unsatisfactory performance rates, expected system behaviour, and assessing uncertainties (Ayyub, 2001). Expert opinion elicitation process is usually performed in a face-to-face meeting. Participants in such a meeting should be informed in advance of the background information, objectives, list of issues, and anticipated outcome. Software engineering researchers have used an expert opinion approach to gain feedback to evaluate and support various models such as requirements process maturity model (Beecham et al., 2005), a ranking model for software engineering measures (Li and Smidts, 2003), and a model of measuring the key factors of success in software process improvement (Dyba, 2000). The reliability of using expert judgment has been demonstrated in various studies. For example, Lauesen and Vinter found that expert's ability to identify requirements defects prevention techniques was very reliable (Lausen and Vinter, 2001). Kitchenham et al. demonstrated that a human centred estimating process incorporating expert opinion can be substantially outperform simple function point models (Kitchenham et al., 2002).

3.1 Research Approach

We sought the experts' opinion on the suitability of FOCSAAM's elements for comparing or assessing different architecture analysis methods as part of a large project aimed at capturing best industrial practices in the area of software architecture analysis. This study uses some of the data gathered during the large project, which itself is not within the scope of this paper. We elicited experts' opinion to assess FOCSAAM using a questionnaire, which was administered to both the participants in two focus group sessions and during face-to-face interview based studies towards the end of the focus group and interview discussions. The respondents were provided with the contextual information on the software architecture analysis topic as part of the invitation to participate in the focus group and interview-based studies. They had discussed the topic in their respective organisational context during the focus group or interview sessions. So by the time they were given the questionnaire to gain their opinion about different elements of the FOCSAAM, they had gained sufficient background knowledge about the need for comparing or assessing different architecture analysis methods.

3.1.1 Research instrument construction

We designed a questionnaire based on the elements of FOCSAAM. The questionnaire had 17 questions; one for each element. Each question was based on the questions included in FOCSAAM under the heading of "brief explanation." The format of each question was a statement based on each of the elements of FOCSAAM and the respondents were required to provide their opinion on whether they agreed or disagreed with the statement in the context of software architecture analysis methods. The responses were sought against a four-point scale (i.e., Strongly agree (SA), Agree (A), Disagree (D), and Strongly Disagree (SD)). The questionnaire also had some space for explaining a particular choice on the scale.

The questionnaire was reviewed for the format of the questions, suitability of the scale, and understandability of the wording. The questionnaire was reviewed by 3 independent researchers and practitioners who had significant experience in the area of software architecture. The reviewers were also asked to check that the time required to respond to all questions was within the range of 10-15 minutes. We could not carry out a formal pilot study to assess the questionnaire because of resource constraints.

3.1.2 Expert selection

In order to elicit expert opinion, the expert should be selected so that the most accurate judgments are provided. Since, it is very difficult to know how exactly this objective can be achieved, it is vital to formulate criteria for

selecting experts (Li and Smidts, 2003). The selection process of selecting experts for our study was governed by the following criteria:

- At least 5 years of software architecture design and analysis in different industrial domains such as finance, medicine, education and retail.
- Willingness to act as impartial evaluator.
- Availability and willingness to commit needed time and effort.
- Specific related knowledge and expertise of the issues of architecture analysis; and
- Willingness to provide effective analysis, and interpretations.

We also ensured that the participants were invited from both in-house software development companies as well as software vendors.

According to the selection criteria, our study needed responses concerning a very specific set of software engineering practitioners, software architects with several years of solid practical experience. Such practitioners usually have time constraints and are not likely to respond to invitation from unfamiliar sources, which makes it hard to apply a random sampling. Consequently, it was decided to use non-probabilistic sampling techniques, availability sampling and snowball sampling. Availability sampling operates by seeking responses from those people who meet the inclusion criteria, if defined, and are available and willing to participate in the research. Snowball sampling means asking the participants of the study to nominate other people who might be willing to participate. The major drawback of non-probabilistic sampling techniques is that the results cannot be considered statistically generalisable to the target population (Kitchenham and Pfleeger, 2001-2002), in this case software architects. However, given the abovementioned constraints, it is believed that the sampling techniques were reasonable.

We used two means of contacting potential respondents: personalised contact and professional referrals. We sent invitation emails to a selected pool of software architects drawn from two sources: practitioners' profile database provided by the industry liaison of Empirical Software Engineering program of the National ICT Australia (NICTA) where this research was carried out, and industry contacts of the researchers. The invitees were also requested to forward the invitation to anyone else who could meet the inclusion criteria. Two of the participants were nominated by the original invitees.

3.1.3 Questionnaire administration

The study involved 17 participants from different organisations. All seventeen software architects had at least five years of software design and several years of software development experience. All the participants had designed and analysed software architectures for large, complex software-based systems in medium to large organisations, which have disciplined and systematic software architecture processes. Ten out of the seventeen participants participated in the focus group discussion and seven were interviewed as part of our large-scale study aimed at identifying the best practices of software architecture analysis. All of them were given questionnaire towards the end of the focus group discussion or interview session

4. FINDINGS AND DISCUSSION

This section presents the findings of the expert-opinion based assessment of FOCSAAM and discusses their implications. The findings are based on the frequencies for each of the responses to each question included in the questionnaire used to obtain experts' opinion about the suitability of different elements of FOCSAAM. The findings indicate that majority of the respondents agreed that all of the elements in FOCSAAM can be used as checklist questions when choosing a suitable architecture analysis method and can be used for comparing and assessing different architecture analysis methods. However, there were a few elements that were not considered important for comparing analysis methods by some of the participants. Before summarizing the frequencies of the responses for each of the elements of FOCSAAM under the headings of its four components in the following sections, this paper presents brief information about the demographics of the respondents.

The respondents' experience in the software development varies between 7 years and 34 years with an average of 18.88 years. On average, they have worked in software architecture design and analysis for 8 years. The average number of projects on which the participants had worked was 32.9. And the average size of their organisations was 12097 employees. This demographic information gives us confidence that the participants of the study were experienced practitioners in the area of software architecture and were working for medium to large organisations.

4.1 Context

The **context** component of FOCSAAM consists of those elements that provide information about a method, for example, what types of goals can be best achieved by using a certain method, which application domains are more appropriate for using a particular method, and which is the most suitable stage of software development lifecycle

for using a given method. The context component comprises six elements (definition of software architecture, method’s goals, number of quality attributes supported, applicable project stage, inputs required and outputs produced, and application domain). Each element of this component can be used to evaluate a software architecture analysis method or to compare different methods. The objective of this component is to help a method’s evaluator determine whether or not a certain method is appropriate for which an architecture analysis is to be performed.

Table 2 shows that there were four elements (i.e., software architecture definition, specific goal, quality attributes, and inputs and outputs) of FOCSAAM which were positively viewed by all the respondents. They either strongly agreed or agreed with their value in terms of stimulating suitable questions to be asked while comparing or assessing architecture analysis methods. However, two elements (i.e., applicable stage and application domain) elicited some disagreement. Two of the respondents did not think that architecture analysis methods should be specific to a certain stage of software development. Rather, they believed that architecture analysis methods should be equally applicable during all stages of the software development lifecycle.

TABLE 2: SUMMARY OF MAIN FINDINGS ABOUT THE CONTEXT COMPONENT.

Elements	Summary of the findings based on responses	SA	A	D	SD
SA definition	A software architecture analysis method should provide an explicit definition of software architecture or it should help define what architecture means.	5	12		
Specific goal	A method should explicitly describe what types of the goals of architecture analysis can be achieved by using that method.	5	12		
Quality attributes	A method’s description should clearly state the quality attributes that can be analysed using that method.	6	11		
Applicable stage	A method should describe the phase of software development lifecycle during which that method is considered most appropriate to use.	5	10	2	
Input & output	A method should explicitly describe the inputs required and outputs produced during architecture analysis.	9	8		
Application domain	A method’s description should mention the domain for which that method is more suitable.	3	10	4	

There are architecture analysis methods which claim to be applicable to any stage of the software development lifecycle, but several methods are considered more appropriate for a certain development stage such as Active Reviews for Intermediate Design (Clements, 2000) ARID and Scenario-Based Architecture Reengineering (Bengtsson and Bosch, 1998). Four respondents disagreed that a method can be domain specific and that a method’s description should mention the most suitable domain for that method. One of these respondents explained that software architecture analysis techniques tend to be common across different domains as the importance of the non-functional requirements spans different domains, and so that is why a method to address issues related to such requirements also needs to be applicable across various domains.

However, architecture analysis methods may be considered more appropriate for domains in which they are validated or most frequently used. We believe that the applicable domain for a method is important because if a method has been frequently used by organisations in a certain domain, and other organisations in that domain need a method to analyse software architecture, they may choose the same method. Moreover, if an architecture analysis method has been used in a number of domains with demonstrable positive benefits, then organisations in those domains will be less reluctant to use that method.

4.2 Stakeholders

The second component of FOCSAAM consists of elements that are mainly of interest to the stakeholders involved in an architecture analysis process. Stakeholders are those who have an interest in the proposed architecture or analysis results such as the designers of the architecture to be analysed, architecture evaluators, initiators of the analysis process, and project managers. Before deciding to use a certain method for analysing an architecture or participating in an analysis exercise, it may be necessary for each kind of the stakeholders to know what benefits can be expected, who should be involved, what process support is available, what socio-technical issues can be expected and what guidance is available to deal with them, and the nature and amount of resources required to use a certain method for analysing a software architecture. These types of questions are included under this component of FOCSAAM. The stakeholder component includes five elements: benefits, stakeholder involvement,

process support, support for socio-technical issues, and resources required. These elements are expected to help stakeholders to assess an architecture analysis method or compare different methods.

Table 3 presents the summary of main findings about the Stakeholders component. Two of the elements (i.e., Stakeholders involvement and Process support) of FOCSAAM were unanimously supported by all participant; they either strongly agreed or agreed that an analysis method should help identify stakeholders that can participate in the architecture analysis process and provide sufficient process support. There were three elements (i.e., benefits, socio-technical issues, and required resources) with which not all the participant agreed. Three of the participants did not agree that an analysis method should explicitly describe the benefits of using the method. One of the respondents elaborated on his response by stating that “benefits of analysing architectures should be covered by the goals of architecture analysis.” This looks to be valid point however, as different methods may be specialised to achieve different goals, which can provide a basis to compare the methods. Similarly, architecture analysis methods also vary on the basis of types of benefits promised and the mechanics of demonstrating the benefits.

Another respondent explained his disagreement by reporting that “instead of a method describing what benefits are provided, an organisation needs to understand why they are analysing architecture and what benefits can be expected.” However, we assert that once an organisation expects certain benefits from analysing architecture, it may like to compare different methods to find out which method promises the benefits that have a close match with what organisation is expecting. A method may also help an organisation to precisely define the benefits it expects and determine the means of assessing the achievement of those benefits.

TABLE 3: SUMMARY OF MAIN FINDINGS ABOUT THE STAKEHOLDER COMPONENT.

Elements	Summary of the findings based on responses	SA	A	D	SD
Benefits	A method should explicitly describe the expected benefits of using that method.	1	13	3	
Stakeholders involvement	A method should help identify stakeholders who are to be involved in the software architecture analysis process.	6	11		
Process support	A method should provide sufficient process support.	1	16		
Socio-technical issues	A method should guide users on dealing with socio-technical issues involved.	3	10	4	
Required resources	A method should explicitly state the types of resources required to analyse architecture.	3	13	1	

Four of the respondents did not think that an architecture analysis method should guide users on dealing with socio-technical issues. One of the respondents commented that “socio-technical issues are organisational dependent and a method needs to know a lot about a particular organisation to provide such support.” While it is true that each organisation would have its own socio-technical issues based on organisational politics and vested interests of different parties involved in analysing architecture, there are nevertheless issues that appear to be common in architecture reviews sessions (such overemphasizing quality goals, egoistic and uncivilised discussions, and hijacking the agenda (Obbink et al., 2001, Kazman and Bass, 2002)). Thus, we believe that architecture analysis methods should be expected to provide some sort of guidance for dealing with these issues, further research needs to be conducted to assess the value of this element of the framework.

Finally, one respondent disagreed that a method should explicitly state the types of resources required to support the software architecture analysis process. That respondent explained his response by stating that “the types of resources required for analysing architecture should be standard across all methods”. We have found that different methods require different types of resources, which also vary depending upon the size of the system, whose architecture is being analysed (Ali-Babar et al., 2004). Thus, we believe that the required resources are a factor in comparing and assessing different architecture analysis methods.

4.3 Contents

The contents component of FOCSAAM is concerned with an architecture analysis method itself. It includes those elements, which help gain a better understanding of the more technical aspects of an analysis method. From a method’s technical point of view, it is necessary to know how it supports the problem-solving process in terms of guidance on the activities to be followed and their sequence, the description of software architecture required, the analysis approaches available in different situations, and what sort of tool support is available to support the process. The contents component consists of four elements (i.e., method’s activities, architectural description, analysis approaches, and tool support). Each of these elements may provide a basis for evaluating an architecture analysis method in terms of its support for each element of this component or compare different methods.

Table 4 presents the summary of main findings about the contents component. None of the elements (i.e., method's activities, architecture description, evaluation approaches, and tool support) of this component were supported by all participants. The majority of the participants were convinced that a method should describe the number and sequence of the activities required to analyse architectures according that method. Three participants disagreed that a method should provide a detailed description of the involved activities as they thought that most of the steps involved in architecture analysis are similar. However, these respondents may not have known that despite having similar activities at the coarse-grained level, different methods vary based on fine-grained details of their activities (Ali-Babar and Gorton, 2004).

The majority of the respondents also supported the need for explicit guidance on the required architecture description language, but three of the respondents did not agree that a method should provide such guidance. They were of the opinion that a method should be independent of how architecture is described as otherwise the method may become too prescriptive. This is a valid point as most of the existing architecture analysis methods are independent of any description language. However, architecture description is one of the most significant inputs for architecture analysis and although most of the methods do not require the use of a certain description language, they do expect architecture description provided in certain formats or in different number of views. Thus, a method should explicitly state the requirements of describing architecture for a certain type of analysis.

TABLE 4: SUMMARY OF MAIN FINDINGS ABOUT THE CONTENT COMPONENT.

Elements	Summary of the findings based on responses	SA	A	D	SD
Method's activities	A method should provide a detailed description of the activities and their sequence to be followed.	5	9	3	
Architecture description	A method should explicitly state, if a certain description language needs to be used to document architecture for analysis.	3	11	3	
Evaluation approaches	A method should help users select an evaluation technique that is most suitable for that method.	3	13	1	
Tool support	A method should be supported by a suitable toolset.	4	8	5	

The last element of this component of FOCSAAM, tool support, was the element with which the largest number (five) of respondents disagreed. Remaining 12 participants thought that appropriate tool support for a method can be a differentiation point. Some of the participants who disagreed with the importance of tool support as a differentiation point elaborated their opinion with these comments: "a methodology should be independent of tools. Tool support for architecture should be same for all methods." While it is a good practice to make a methodology independent of a tool that can support different tasks of that methodology, certain aspects of a methodology may need very specific features that might not be provided by generic tools.

For example in ATAM, quality attributes are specified by building a utility tree, and results can be presented using a result tree, however, other methods do not use these techniques. Thus, any tool that claims to sufficiently support ATAM is expected to help build utility and result trees. Furthermore, if an architecture analysis method requires architecture be described in a certain description language (such as UniCon, Acme, Wright, and Rapide), that method should have a supporting tool technology to help create, maintain, evolve, and analyze the architectures specified in the required architecture description language. Moreover, a recent effort to assess three architecture analysis methods using the features analysis approach considered tool support provided by each method as an assessment criterion (Griman et al., 2006). FOCSAAM compares architecture analysis methods based on the level of support provided by a tool. Such support may vary from non-existent to full support. However, there is a need for further research on the criticality of using tool support as a differentiation point for architecture analysis method.

4.4 Reliability

The last component of FOCSAAM comprises two elements, maturity of a method and a method's validation, which are critical aspects to be considered in the selection of any technology. The answers to the questions regarding these two elements may affect an evaluator's perception of a method's reliability. A more mature and rigorously validated method is considered more reliable than the one that is relatively new and informally validated.

Table 5 shows the summary of main findings about the Reliability component, which has only two elements: maturity of method and method's validation. A majority of the respondents supported the role of a method's maturity in comparing different architecture analysis methods. There was one respondent who disagreed with this statement without providing any explanation. The second element of this component, method's validation,

was supported by all the respondents. These findings provide justification for the inclusion of both elements of the reliability component of FOCSAAM and these elements can form a basis for comparing and assessing software architecture analysis methods.

TABLE 5: SUMMARY OF MAIN FINDINGS ABOUT THE RELIABILITY COMPONENT.

Elements	Summary of the findings based on responses	SA	A	D	SD
Maturity of method	A method's maturity is an important factor for selecting a particular method.	2	14	1	
Method's validation	Validation of the method undertaken by its developer is an important factor in choosing a certain method.	3	14		

5. SUMMARY AND CONCLUSION

The goal of this research was to assess the suitability of different elements of a framework for comparing software architecture analysis methods. To achieve this aim, this research used the expert opinion approach, which asked practicing architects to assess the suitability of the elements of FOCSAAM as checklist questions to be asked while selecting a particular method to support the software architecture analysis process. The expert opinion based assessment of FOCSAAM revealed that a majority of the participants agreed with the suitability of most of the elements of FOCSAAM as a basis for comparing and assessing software architecture analysis methods. There were seven elements (i.e., software architecture definition, specific goal, quality attributes, inputs and outputs, stakeholder involvement, process support, and method's validation), which were supported by all the respondents. Other elements were supported by a majority of the participants with some disagreements on the value of certain elements (such as tool support, method's activities, and application domain) as differentiation points for comparing analysis methods. However, the participants did not provide detailed explanation for their disagreement as to why certain elements of FOCSAAM cannot be a differentiation point.

Lack of explanation makes it difficult to understand the rationale for participants' disagreement. However, we can offer at least two explanations for lack of support for certain elements of FOCSAAM among the participants. First, the wordings of the questions included in the questionnaire might not have been clear enough to fully understand the purpose and meaning of a certain question based on a particular element of FOCSAAM. For example, three of the participants disagreed with the statement that "*A method should explicitly state, if a certain description language needs to be used to document architecture for analysis.*" It would be interesting to know whether or not the responses would have been different if we had reshaped that statement in these words "*If there are constraints on analysis process as a result of the format of architecture description, the method should clearly report those constraints.*" Second, the participants might have chosen to identify some elements of FOCSAAM less critical than the other elements. However, our study did not seek the participants' opinion about the relative criticality of different elements of FOCSAAM. For example, a significant number of the participants disagreed with the tool support element's ability as a differentiation point among software architecture analysis methods. One interpretation of this response can be that tool support can be part of architecture analysis method but perhaps less critical factor.

There are certain limitations that need mentioning: 1) we were unable to determine and understand the bias that might have been existed in experts' inputs; 2) we did not have a large sample size. Neither of these limitations can be easily removed as availability of resources would be barrier to any similar study.

Based on the findings of this research, there is sufficient support for each element of FOCSAAM to be included in forming criteria for comparing or assessing architecture analysis methods. There is a need for future research that will allow us extend the assessment process to a wider range/number of experts. Moreover, the experts should also be asked if there are other elements that should be included in FOCSAAM.

6. ACKNOWLEDGMENTS

We are grateful to the participants of this study. Len Bass and Ian Gorton facilitated the focus group sessions. Authors were working with National ICT Australia when the reported research was carried out and this manuscript was produced.

REFERENCES

ABOWD, G., BASS, L., CLEMENTS, P., KAZMAN, R., NORTHROP, L. & ZAREMSKI, A. (1997) Recommended Best Industrial Practice for Software Architecture Evaluation. Software Engineering Institute, Carnegie Mellon University.

- ALI-BABAR, M. & GORTON, I. (2004) Comparison of Scenario-Based Software Architecture Evaluation Methods. *Proceedings of the 1st Asia-Pacific Workshop on Software Architecture and Component Technologies*. Busan, South Korea.
- ALI-BABAR, M., ZHU, L. & JEFFERY, R. (2004) A Framework for Classifying and Comparing Software Architecture Evaluation Methods. *Proceedings of the 15th Australian Software Engineering Conference*. Melbourne, Australia.
- AYYUB, B. M. (2001) A Practical Guide on Conducting Expert-Opinion Elicitation of Probabilities and Consequences for Corps Facilities. Institute for Water Resources, Alexandria, VA, USA.
- BASS, L., CLEMENTS, P. & KAZMAN, R. (2003) *Software Architecture in Practice*, Addison-Wesley.
- BEECHAM, S., HALL, T., BRITTON, C., COTTEE, M. & RAINER, A. (2005) Using an expert panel to validate a requirements process improvement model. *Journal of Systems and Software*, 76, 251-275.
- BENGTSSON, P.-O. & BOSCH, J. (1998) Scenario-based Architecture Reengineering. *5th International Conference on Software Reuse*.
- CLEMENTS, P., KAZMAN, R. & KLEIN, M. (2002) *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley.
- CLEMENTS, P. C. (2000) Active Reviews for Intermediate Designs. SEI, Carnegie Mellon University.
- DOBRICA, L. & NIEMELA, E. (2002) A Survey on Software Architecture Analysis Methods. *IEEE Transactions on Software Engineering*, 28, 638-653.
- DYBA, T. (2000) An instrument for measuring the key factors of success in software process improvement. *Empirical Software Engineering*, 5, 357-390.
- FORSELL, M., HALTTUNEN, V. & AHONEN, J. (1999) Evaluation of Component-Based Software Development Methodologies. IN PENJAN, J. (Ed.) *Proc. of FUSST*. Tallinn.
- GRIMAN, A., PEREZ, M., MENDOZA, L. & LOSAVIO, F. (2006) Feature analysis for architectural evaluation methods. *Journal of Systems and Software*, 79, 871-888.
- JAYARATNA, N. (1986) Normative Information Model-Based Systems Analysis and Design (NIMSAD): A Framework for Understanding and Evaluating Methodologies. *Journal of Applied Analysis*, 13, 73-87.
- KAZMAN, R. & BASS, L. (2002) Making Architecture Reviews Work in the Real World. *IEEE Software*, 19, 67-73.
- KAZMAN, R., BASS, L., KLEIN, M., LATTANZE, T. & NORTHROP, L. (2005) A Basis for Analyzing Software Architecture Analysis Methods. *Software Quality Journal*, 13, 329-355.
- KITCHENHAM, B., PFLEEGER, S., MCCOLL, B. & EAGAN, S. (2002) An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software*, 64, 57-77.
- KITCHENHAM, B. & PFLEEGER, S. L. (2001-2002) Principles of Survey Research, Parts 1 to 6. *Software Engineering Notes*.
- LASSING, N., RIJSENBRIJ, D. & VLIET, H. V. (1999) The goal of software architecture analysis: Confidence building or risk assessment. *Proceedings of First BeNeLux conference on software architecture*.
- LAUSEN, S. & VINTER, O. (2001) Preventing requirement defects: an experiment in process improvement. *Requirements Engineering Journal*, 6, 37-50.
- LI, M. & SMIDTS, C. S. (2003) A ranking of software engineering measures based on expert opinion. *IEEE Transactions on Software Engineering*, 29, 811-824.
- MATINLASSI, M. (2004) Comparison of Software Product Line Architecture Design Methods: COPA, FAST, FORM, Kobra and QADA. *Proc. of the 26th Int'l Conf. Software Eng.* Edinburgh, Scotland.
- OBBINK, H., KRUCHTEN, P., KOZACZYNSKI, W., POSTEMA, H., RAN, A., DOMINICK, L., KAZMAN, R., HILLIARD, R., TRACZ, W. & KAHANE, E. (2001) Software Architecture Review and Assessment (SARA) Report. SARA W.G.

Appendix A

The questionnaire used for this study cannot fit within the limit of 10 pages that is why we have made it available from the following link:

<http://www.cse.unsw.edu.au/~malibaba/Questionnaire.pdf>