M. Svahnberg and C. Wohlin, "A Comparative Study of Quantitative and Qualitative Views of Software Architectures", Proceedings EASE: Empirical Assessment and Evaluation in Software Engineering, Keele, UK, 2003.

A Comparative Study of Quantitative and Qualitative Views of Software Architectures

Mikael Svahnberg and Claes Wohlin

Department of Software Engineering and Computer Science
Blekinge Institute of Technology, PO Box 520, S-372 25 Ronneby SWEDEN
[Mikael.Svahnberg | Claes.Wohlin]@bth.se, http://www.ipd.bth.se/serl

Abstract. *In order to obtain a software architecture with the right quality attributes, it is vital to fully understand the benefits and liabilities of all involved architecture candidates. To this end, all possible sources should be used. In this paper we compare a quantitative description of software architectures and the support given for different quality attributes with a qualitative description. In some areas they strengthen each other, but in others they do not. In conclusion, this study shows a need to increase the understanding of the quality strengths and weaknesses of different software architectures.*

1 Introduction

When developing software, it is important to have an appropriate architecture for the system, or sub-systems comprising the full system. There are many factors influencing the choice of, or evolution into, an appropriate software architecture. For example the process of creating a software architecture is not only governed by functional requirements but to a large extent by quality attributes as indicated by e.g. [Bass et al. 1998][Bosch 2000][Hofmeister et al. 2000].

One way to design a software system and to ensure certain quality attributes is to start with an architecture style [Buschmann et al. 1996][Shaw & Garlan, 1996]. These architectural styles typically have certain documented qualities, both positive and negative.

However, it is still a non-trivial task to discern between the architectures as described in literature today (e.g. [Buschmann et al. 1996]). Although there are benefits and liabilities listed, there is no order between them or any measure of how good or bad the architectures are for each of the quality attributes. This, in turn, makes it difficult to compare the benefits and liabilities of different architectures.

Moreover, the documented benefits and liabilities of the software architectures may not be relevant or have the same priority for the domain of the software system in question, i.e. the context may differ from where the application is intended to run. This means that the architectures need to be augmented with the experience of the software developers, and hence decisions about which software architecture style to use or strive for are often taken based on the intuition of senior software developers.

To avoid basing such crucial decisions on only the intuition and subjective judgements of senior software developers, it is important to be able to quantify the knowledge of developers and to compare software architecture candidates for a system to build based on quantified data as well as the qualitative literature view. This should act as a complement to intuitive judgements involved when selecting between architecture candidates. Moreover, a method for quantifying knowledge creates a learning effect, where new software developers can learn from the structured knowledge of the senior software developers.

Our hypothesis is that there are different views of the benefits and liabilities of different software architecture candidates. Based on their respective background and experiences, subjects will come to different conclusions of the benefits and liabilities of software architectures, and these conclusions may not concur with e.g. the predominant literature view.

To study this we present, in this paper, a comparison of two different descriptions of software architecture styles, one based on quantitative data based on subjective judgements and relative comparisons and one based on a qualitative description as typically found in literature. This comparison is performed to provide a new perspective on the qualitative descriptions usually found in literature using another data source with a unique background and set of experiences. The results from this comparison may be such that they increase the confidence when using either, or both, of the descriptions, or they may indicate where there is a need to increase the understanding of the quality strengths and weaknesses of different software architectures.

The remainder of this paper is organized as follows. In Section 2 we describe the background of this study and the techniques we use. In Section 3 we compare a qualitative view of software architectures and their quality attributes with a quantitative view. The comparison is analysed in Section 4 and the paper is concluded in Section 5.

	Microkernel	Blackboard	Layered	Model-View-Controller	Pipes and Filters
Efficiency	0.161	0.145	0.0565	0.0557	0.218
Functionality	0.119	0.321	0.237	0.115	0.151
Usability	0.106	0.127	0.255	0.104	0.0818
Reliability	0.122	0.0732	0.0930	0.105	0.144
Maintainability	0.183	0.273	0.221	0.300	0.271
Portability	0.309	0.0597	0.138	0.320	0.135

Table 1: Framework for Architecture Structures (FAS)

	Microkernel	Blackboard	Layered	Model-View-Controller	Pipes and Filters
Efficiency	0.264	0.175	0.0868	0.113	0.360
Functionality	0.205	0.252	0.199	0.206	0.139
Usability	0.0914	0.113	0.250	0.408	0.137
Reliability	0.126	0.142	0.318	0.190	0.224
Maintainability	0.191	0.0921	0.285	0.239	0.193
Portability	0.112	0.0689	0.426	0.139	0.255

Table 2: Framework for Quality Attributes (FQA)

2 Background

In [Svahnberg & Wohlin 2002a] we describe an experiment where eight experts in software engineering completed a questionnaire based on a method called Analytic Hierarchy Process [Saaty 1980][Saaty & Vargas 2001] (described below) in order to construct a quantitative framework of the benefits and liabilities of different architecture candidates with respect to a set of quality attributes.

This framework consists of two tables which we refer to as a *framework for architecture structures* (FAS) and a *framework for quality attributes* (FQA). The FAS rates the potential support each architecture candidate can give for the different quality attributes, and the FQA ranks which architecture candidate is best at each quality attribute. Table 1 and Table 2 present these two tables from the experimental study. The tables are constructed from pairwise comparisons of each architecture candidate for all quality attributes (Table 1) and for each quality attribute for all architecture candidates considered (Table 2). The pairwise comparison is done using the Analytic Hierarchy Process (AHP). The colouring (grey) of the cells indicate how the comparison has been done. The values assigned are the relative importance and they sum to one in columns in Table 1 and in rows in Table 2. The Analytic Hierarchy Process is described briefly below.

It is worth noting with Table 1 and Table 2 that they are normalised so that the columns in Table 1 and the rows in Table 2 sum up to one. Hence, the values are *relative comparisons*, which means that what can be extracted from the frameworks is a ratio of how much better one architecture is than another for a particular quality attribute. What cannot be extracted is an absolute number of how good an architecture is. For example, portability scores 0.309 for Microkernel in Table 1. This means that it is the quality attribute that Microkernel supports best of all the studied attributes, e.g. almost thrice as well as usability, functionality and reliability (scoring 0.106, 0.119 and 0.122, respectively). On the other hand, in Table 2 Microkernel only gets a value of 0.112 for portability, which means that three other architectures (i.e. Layered, Pipes and Filters and Model-View-Controller) are better at portability than Mircokernel. In fact, the score for Microkernel is only better than that of Blackboard. Moreover, we see that Layered (with a score of 0.426) is perceived as almost four times as good at supporting portability than Microkernel.

Analytic Hierarchy Process. The Analytic Hierarchy Process (AHP) has previously been described, evaluated and successfully used in similar settings and in other areas of software engineering (e.g. [Karlsson & Ryan 1997][Karlsson et al. 1998][Shepperd et al. 1999]). Briefly, AHP consists of a set of steps, where all combinations of elements are evaluated pair-wise, and according to a certain scale, as illustrated in Figure 1. The question to answer for each pair-wise comparison is which of the two elements, *i* or *j* is more important, and how much more important it is. This is rated by interpreting the values as presented in Table 3. The questionnaire for an AHP ses-

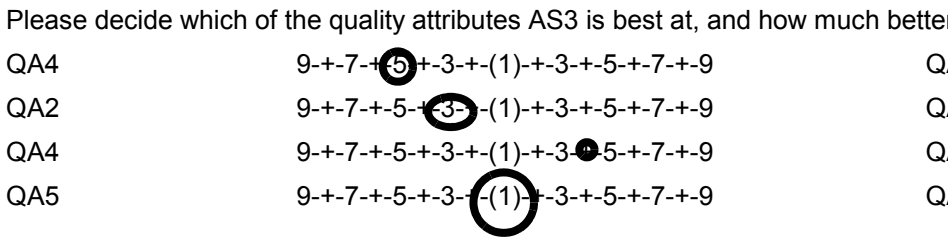

Figure 2. An Example of an AHP Questionnaire

sion is thus fairly simple, as the extract from an AHP questionnaire in Figure 2. In this figure, the five quality attributes QA1 to QA5 are compared for architecture candidate AS3. The figure also illustrates how to complete the questionnaire by circling the desired alternative.

These comparisons are then transferred into a $n \times n$ matrix, where n is the number of elements, together with the reciprocal values. After this is done, the eigenvector of the matrix is computed. Saaty [Saaty 1980][Saaty & Vargas 2001] proposes a method called *averaging over normalized columns* to do this. This results in an estimation of the eigenvalues of the matrix, and is called the *priority vector*. The priority vector is the primary output of applying AHP.

Next, the fact that AHP uses more comparisons than necessary (i.e. $n \times (n-1)/2$ comparisons) is used to evaluate the consistency of the rating. A consistency ratio *(CR)* is calculated, which indicates the amount of contradictions and inconsistencies between the pair-wise comparisons. A consistency ratio of 0.10 or less is, according to [Saaty 1980][Saaty & Vargas 2001], considered acceptable even if it is pointed out that higher values are often obtained. Hence 0.10 may be too hard, but it nevertheless indicates an approximate size of the consistency ratios to expect.

A more extensive description of AHP can be found in e.g. [Saaty 1980][Saaty & Vargas 2001][Karlsson et al. 1998] and [Svahnberg & Wohlin 2002a].

We found AHP to be useful for our needs, as it is an accepted method from management science that enables a quantification of subjective judgements that can be hard to capture otherwise. Moreover, since it provides a consistency ratio we are able to determine how much trust we can put in each individual's answers.

It should kept in mind that AHP only collects and summarises subjective judgements. As such, the results are vulnerable to the usual problems of subjectivity, in that the participants form their own interpretations of the factors and the evaluation scale. Our approach to address this is to use several subjects and then use the mean value of these when conducting the comparison with literature.

Moreover, as we state in our hypothesis, part of the intention of this paper is to investigate if there are differences between perceived benefits and liabilities of different software architectures. The frameworks described above represent one view based on the participants' personal backgrounds and experiences of the software archi-

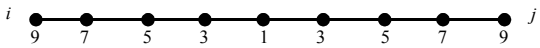

Figure 1. The scale for the AHP comparison.

Relative intensity	Definition	Explanation
1	Of equal importance	The two variables (*i* and *j*) are of equal importance.
3	Slightly more important	One variable is slightly more important than the other.
5	Highly more important	One variable is highly more important than the other.
7	Very highly more important	One variable is very highly more important than the other.
9	Extremely more important	One variable is extremely more important than the other.
2, 4, 6, 8	Intermediate values	Used when compromising between the other numbers.
Reciprocal	If variable *i* has one of the above numbers assigned to it when compared with variable *j*, then *j* has the value 1/number assigned to it when compared with *i*. More formally if $n_{ij} = x$ then $n_{ji} = 1/x$.	

Table 3: Scale for pairwise comparison using AHP [Saaty 1980][Saaty & Vargas 2001].

tectures. Hence, the subjectivity of AHP (i.e. that the experience and background of the subjects play a part in the result) is in fact beneficial for our purposes.

3 Comparing Literature and a Quantitative Study

There is a number of factors to consider when comparing different approaches to describing software architectures.

First and foremost is the fact that different formats for describing the architectures can be used (as is the case in the comparison in this paper), and to make a comparison one has to make interpretations and translations between these descriptions.

Another factor to consider is that the focus of the descriptions are different. Whereas the literature typically provide alerts and issues to consider when using a particular software architecture (see e.g. [Buschmann et al. 1996] and [Bosch 2000]), a quantitative description provides a means for comparing how well different quality attributes are met, i.e. how important the benefits and liabilities described by the literature really are.

Moreover, the quantitative framework in this study is constructed in a way that allows for comparing different architecture candidates based on different sets of desired quality attributes. These architecture candidates and quality attributes may be different every time the process is executed. This comparison is usually harder to find in the literature, which typically only enables a qualitative comparison of the quality attributes within a certain software architecture, and not between architecture candidates.

The comparison in this section is conducted by comparing the FAS and FQA presented in Table 1 and Table 2, which are a combination of the opinions of eight domain experts from academia, with what pertinent literature identifies as strengths and weaknesses.

The quantitative view is hence represented by the FAS and FQA, and the qualitative view is represented by the literature. The literature view focus on [Buschmann et al. 1996] and [Bosch 2000]. We use [Buschmann et al. 1996] since the architectures in the quantitative study are those presented in this book, and [Bosch 2000] because of a, in our opinion, relatively good section on the strengths and weaknesses of different software architectures. Moreover, both of these books are relatively well known and have made an impact on the software architecture community.

The purpose of this study is to investigate whether there are differences between different views of software architectures. To show this it is not necessary to conduct an exhaustive comparison between all views known to date of different software architectures. Instead, we focus on the two aforementioned sources in contrast to the study we have performed ourselves.

The FAS and FQA are created using subjective judgements. Together with the fact that the architectures used in the quantitative study are the same as those in [Buschmann et al. 1996] this introduces a risk that the subjects participating in the study may have gained their knowledge of the architectures from the very same literature. This would mean that the two data sources are not, in fact, independent. In Table 4 we present the experiences of the participants in the quantitative study, to allay this threat. As can be seen in this table, all participants have experiences that extend beyond those of just reading the literature in this study. This, of course, does not rule out that they have been influenced by what can be found in [Buschmann et al. 1996] and [Bosch 2000] - it just ensures that their experiences include more than those two sources and that they have practical experience with the architectures as well as theoretical knowledge.

It should be noted that neither [Buschmann et al. 1996] nor [Bosch 2000] use the same categorization of quality attributes as in our study (i.e. [ISO 9126]), wherefore a certain amount of translation has been made. Moreover, [Bosch 2000] does not present all of the architectures used in our study.

As the framework, quantitative in nature, is meant to complement the qualitative information found in [Buschmann et al. 1996] and [Bosch 2000], we are forced to use subjective interpretations of the literature in order to make a comparison. Therefore, it should be noted that the comparison below, and indeed any comparison between quantitative frameworks and qualitative literature (at least the literature used in this study), is done using a subjective interpretation of the information extracted from literature.

There are cases where the literature does not mention a particular quality attribute, neither as a benefit nor as a liability for a certain architecture. In these cases, we have interpreted this to mean that the literature sees nothing extraordinary with the architecture with respect to the not mentioned quality attribute, i.e. it is neither a benefit nor a liability. The quantitative framework should hence in these situations list the quality attribute in a similar fashion, i.e. neither with very high nor with very low values. Another issue is when the literature mentions both benefits and liabilities of an architecture with respect to a particular quality attribute. This may either indicate that the quality attribute is specified on too high a level of abstraction and needs to be broken down into sub-factors instead, or that it depends on the context where the architecture is used whether it will be good or bad at the quality attribute.

Name	Title	Experience
Bob	PhD Student	Have done research on software development together with industry partners. Have participated in several large development projects.
Larry	Professor	Some industry practise. Have done research on conflicts between quality attributes in software systems together with industry partners. Have done research on several large industry applications, involving their software architecture.
Edward	PhD	Have done research related to specific quality attributes. Have experience supervising development projects. Have been teaching object-oriented design methods and quality attributes in object-oriented design.
Kenneth	PhD	Have done research on software architectures together with industry partners. Have done research on software product-line architectures together with industry partners. Have done research on object-oriented frameworks together with industry partners.
Ivan	PhD	Several years of industry practise. Part-time industry employed. Have done research on conflicts between quality attributes in software systems together with industry partners.
Nathan	PhD Student	PhD studies specialised in software architectures and architecture evaluation, conducted together with industry partners. Have participated in several large development projects. Have conducted numerous software architecture evaluations together with industry partners.
George	Professor	Several years of industry practise. Have done research on software evaluations and software architecture evaluation together with industry partners.
Eric	PhD	Several years of industry practise. Have done research on software architecture evaluation. Have done research on software architectures together with industry partners. Have participated in several large development projects. Have conducted numerous software architecture evaluations together with industry partners.

Table 4. Experiences of Subjects in Quantitative Study

Below, we go through the different architectures in our study and discuss for each quality attribute in our study whether the literature lists this quality attribute as a benefit or a liability. We also provide a brief summary for each architecture with pointers to the areas where more discussion is needed.

3.1 Microkernel

Efficiency. The quantitative study suggests that microkernel is fairly good at efficiency, especially compared to other architectures (in the FQA), whereas Buschmann suggests that, because of the indirect method calls, performance is a liability but that this can be overcome if designed properly.

Usability. Discussions with participants reveal that the interpretation of this quality attribute has been ease of applying and using the architecture in question. Microkernel scores low on usability in the quantitative study, and this is in conformance with Buschmann which lists complexity of design and implementation as a liability.

Reliability. Listed as a benefit by Buschmann, whereas the quantitative framework suggest that there is nothing extraordinary about microkernel with respect to reliability.

Maintainability. Mentioned among the benefits of microkernel by Buschmann. The quantitative study suggests that it is fairly good at maintainability, but not when compared to the other architectures.

Portability. Listed as a benefit by Buschmann, which conforms partly to the quantitative study which scores microkernel high on portability, albeit not when comparing with other architectures.

Summary. Summarizing microkernel, we have one point where the quantitative framework and literature strongly agree, i.e. usability, and two more just in agreement, i.e. portability and maintainability. However, we also have two points where the quantitative framework and literature disagree, i.e. reliability and efficiency.

It is important to seek an answer to why the opinion of reliability and efficiency differs between the quantitative and qualitative view. For efficiency, the participants in the quantitative study may have made the interpretation that the impact of indirect method calls is not as severe as Buschmann implies. For reliability, there can be several explanations to why the quantitative framework differs compared to literature. For example, the benefit may not be very large compared to the other traits of microkernel.

In summary, the comparison shows that there is a need to increase the understanding of the microkernel with respect to some quality attributes.

3.2 Blackboard

Efficiency. Listed as a liability by both Buschmann and Bosch, which conforms with the quantitative study.

Usability. Not mentioned by literature, and the FAS places usability rather in the middle, i.e. not extremely high nor extremely low. The FQA, however, places blackboard fairly low on usability, but the number is still pretty high, i.e. blackboard is not extremely bad compared to the other architectures. Hence, one can interpret this to imply that the quantitative and qualitative views are in reasonable agreement.

Reliability. Fault tolerance and robustness are listed as benefits by Buschmann, whereas the quantitative study suggests that the reliability of blackboard is not very impressive. Both of these views are confirmed by Bosch.

Maintainability. Listed as a benefit by Buschmann and Bosch, and the quantitative study suggests that it is the second most fulfilled quality of blackboard. However, when compared to the maintainability of other architectures, it scores fairly low.

Portability. Not mentioned by literature, whereas the quantitative study suggests that blackboard is very bad at portability. That literature does not mention portability can be interpreted as that the authors view blackboard as being pretty average on portability.

Summary. There is one point where the quantitative framework and literature strongly agree, i.e. efficiency, and two points in agreement, i.e. maintainability and usability. There is one point where the framework and literature disagree, i.e. reliability. There is also one point where there is a strong disagreement, namely portability.

For portability, the interpretation is of course skewed as it is not mentioned at all by literature, but it is important to find out why it is not mentioned, and whether it matches the quantitative view of portability. For reliability, we see that there are two views, although both of these views are confirmed by one of the qualitative sources. One possible explanation is that the participants of the quantitative study and Buschmann have focused on one of the two views, neglecting the other view.

It is clear from the study that there are some quality attributes that are not fully understood for the blackboard.

3.3 Layered

Efficiency. Listed as a liability by Buschmann, Bosch and the quantitative study.

Usability. Most of the benefits listed by Buschmann are related to the ease of using the layered style, but there are also two points listed as liabilities pertaining to the usability of layered. The quantitative study suggests that layered is good at usability, also when compared to the other architectures.

Reliability. Bosch presents a negative view of the reliability of layered, but also a positive view of the added possibilities for data checking at many layers. The quantitative study suggests that layered is not very good at reliability, but that it nevertheless is better at it than all other architectures in our study.

Maintainability. Both Buschmann and Bosch are fairly positive about the maintainability of layered, which conforms with the quantitative study.

Portability. Listed as a benefit by Buschmann, which the quantitative study also suggests, especially when comparing with the other architectures (in the FQA).

Summary. Layered seems to be a very well understood software architecture. The quantitative framework agrees with the literature in all cases. There is one point in agreement, i.e. reliability, and four points strongly in agreement, i.e. efficiency, usability, maintainability and portability.

3.4 Model-View-Controller

Efficiency. Two of the liabilities mentioned by Buschmann are concerned with efficiency, which conforms to the data from the quantitative study, which ranks model-view-controller low in both the FQA and the FAS.

Usability. Buschmann lists mostly benefits regarding usability, which conforms to the quantitative study where model-view-controller is ranked highest on usability compared to other architectures. However, Buschmann also

lists some negative aspects regarding usability. This is, we believe, indicated in the quantitative study by the fairly low score on usability in the FAS.

Reliability. Not mentioned by literature, and the quantitative study implies that it is neither the strong nor the weak spot for model-view-controller. Hence, the quantitative study and literature seems to be in accord.

Maintainability. For maintainability, Buschmann lists a number of issues as benefits, but also a number of issues as liabilities. It is our opinion that the liabilities outweigh the benefits, which goes against the quantitative study, where maintainability is ranked highly in both tables. However, as Buschmann also lists positive aspects, this disagreement does not seem very large.

Portability. As with maintainability, Buschmann lists both benefits and liabilities with respect to portability. Again, we feel that the liabilities are more severe than the potential benefits, which again goes against the quantitative study. As with maintainability, the severity is lessened by the fact that Buschmann also lists positive aspects.

Summary. For model-view-controller, there are three points where the quantitative framework and the qualitative literature strongly agree, i.e. efficiency, usability and reliability, and two points where they disagree, i.e. maintainability and portability.

The participants in the quantitative study may have made different interpretations of the implications of the benefits and liabilities of maintainability and portability than we have done, which may explain these two disagreements. In summary, there are some quality attributes that need to be better understood for the Model-View-Controller

3.5 Pipes and Filters

Efficiency. Efficiency is listed as a benefit by Buschmann, even if a number of concerns are listed as liabilities as well. More or less the same benefits and liabilities are also presented by Bosch. The quantitative study presents pipes and filters as good at efficiency, both compared to other quality attributes (FAS) and compared to other architectures (FQA).

Usability. Not mentioned in literature. The FAS lists usability as the weakest point of pipes and filters, whereas the FQA lists pipes and filters to be between bad to medium on usability.

Reliability. Listed as a liability by both Buschmann and Bosch, whereas the quantitative study reports pipes and filters as being fairly good at reliability compared to the other architectures. However, compared with other quality attributes, reliability is not this architecture's strong side.

Maintainability. Maintainability issues are listed as benefits by Buschmann, and Bosch agrees. However, Bosch argues that there are liabilities with maintainability as well. The quantitative study indicates that maintainability is the best supported quality attribute, but that the architecture is only moderately good at maintainability compared to the other architectures.

Portability. Not mentioned in literature. The FAS lists portability as a fairly weak point, although not as bad as usability. In the FQA, however, pipes and filters scores very high and is, in fact, the second best architecture in our study with respect to portability.

Summary. For pipes and filters, there is one point where the quantitative framework and literature agree, i.e. maintainability, and another point strongly in agreement, i.e. efficiency. There are two points where the quantitative framework and literature disagree, i.e. usability and reliability, and one point strongly in disagreement, i.e. portability.

The comparison for usability and portability may be an interpretation construct, as these are not mentioned by literature. If this is the case, however, it is interesting to note that the literature fails to mention what the participants in the quantitative study considered to be two of the more important characteristics of pipes and filters.

For reliability, it is possible that the difference between the quantitative and qualitative view comes from that the qualitative view does not compare the architectures with each other, but only look within one architecture and compare the quality attributes. If this is so, the quantitative and qualitative view agrees with each other. Pipes and Filters is a well-known software architecture. However, there is still need for investigating its support for some quality attributes.

4 Analysis and Interpretation

There are basically two outcomes of the comparison between literature and the quantitative study. Either there is a reasonable match between literature and the framework, or there is a mismatch.

If there is a match, this means that we can trust the quantitative framework more and may use it to compare between the architecture candidates. This enables us to find out e.g. which architecture candidate is most suitable for a particular mix of quality requirements. It also means that the views in literature have been confirmed, thus increasing the confidence in this view.

	Microkernel	Blackboard	Layered	Model-View-Controller	Pipes and Filters
Points Strongly in Agreement	Usability	Efficiency	Efficiency Usability Maintainability Portability	Efficiency Usability (Reliability)	Efficiency
Points in Agreement	Portability Maintainability	Maintainability (Usability)	Reliability		Maintainability
Points in Disagreement	Reliability Efficiency	Reliability		Maintainability Portability	Reliability (Usability)
Points Strongly in Disagreement		(Portability)			(Portability)

Table 5: Conformance between the example framework and literature

If the framework and literature do not match, we need to determine whether the perceptions of the creators of the quantitative framework are wrong or if it is the view presented in the literature that is not applicable for the domains envisioned by the participants in the quantitative study. In either case, the understanding of the architecture candidates increases.

In the literature comparison in this paper, we get the results presented in Table 5. In this table we present for each architecture for which quality attributes there is a strong agreement, an agreement, a disagreement or a strong disagreement between literature and the framework used.

We see that while there are many points where the framework concur with literature, there are also several places where the framework contradicts the literature (Albeit sometimes indirectly, as the quality attributes are in many cases not mentioned by the literature. We list these within parentheses in the table.). In these cases it is important to seek an answer to why there is a contradiction. If answers are not found important aspects may be overlooked, which may cause problems as the architectures and the descriptions of them (both the qualitative and quantitative) are used during development of a software system.

The literature is used as an additional source to complement the views expressed quantitatively in the framework and to identify where there is disagreement between literature and the framework or where the framework and literature do not overlap completely, i.e. where there are aspects covered by the one and not by the other. Examining these areas further enables us to seek a deeper understanding of the architectures and quality attributes involved in the literature and the quantitative framework. To seek explanations for the disagreements between the two views, and to extend the comparison to other views as well, would indeed be an interesting future study, albeit a rather massive undertaking.

Connecting back to our hypothesis, we see that the difference between the view represented by literature and the view based on the personal experiences and backgrounds of a set of researchers is large enough to motivate this study. It is not the case that a singe source present an absolute, objective and unambiguous view. Background and previous experiences do play a part when assessing the usefulness of different software architecture candidates.

5 Summary and Conclusions

Software quality is never added as an afterthought. Instead, it is the early design work that ultimately determines what potential a software system will have for fulfilling its quality requirements. Hence, the choice of which software architecture to use is too important to allow for important aspects to be forgotten, or to use architecture styles that are not suitable for the application domain.

In order to determine between a set of architecture candidates it is not enough to use qualitative descriptions of the benefits and liabilities of each candidate. It is also necessary to get some quantitative feel for the different candidates and then relate this to the qualitative descriptions. This ensures that the decision of which architecture candidate to base a software system on is taken based on the best available information.

In this paper we present a comparison between a quantitative view and a qualitative view of software architectures and quality attributes. This comparison allows us to identify where different views (i.e. the quantitative and qualitative views in this paper) have different opinions of the benefits and liabilities of software architectures. If this is not done there is an imminent risk that important aspects are overlooked or ignored and hence that software projects using these descriptions will fail in their goals.

The quantitative framework used in this study have a number of benefits over a qualitative description:

- It allows for a comparison of different architecture candidates.
- It enables the architectures to be assessed with respect to the support given for different quality attributes.
- A new framework can easily be created to use in a particular domain, where other software architectures and quality attributes are relevant.

The comparative study shows that there is still some way to go before a well-established understanding of different software architectures and their respective support or lack of support for different quality attributes exist. Further work is needed with respect to quality attributes and how they are supported by different software architectures, as the study in this paper indicates a lack of understanding in this respect. One path forward is to study comparisons between different sources of architecture descriptions and different studies (such as the quantitative study used in this paper) and seek explanations for where these different sources have different views of the benefits and liabilities of different software architectures.

References

[Anderson et al. 2000] D.R. Anderson, D.J. Sweeney, T.A. Williams, *"An Introduction to Management Science: Quantitative Approaches to Decision Making"*, South Western College Publishing, Cincinnati Ohio, 2000.

[Bass et al. 1998] L. Bass, P. Clements, R. Kazman, *"Software Architecture in Practice"*, Addison-Wesley Publishing Co., Reading MA, 1998.

[Buschmann et al. 1996] F. Buschmann, C. Jäkel, R. Meunier, H. Rohnert, M. Stahl, *"Pattern-Oriented Software Architecture - A System of Patterns"*, John Wiley & Sons, Chichester UK, 1996.

[Bosch 2000] J. Bosch, *"Design & Use of Software Architectures - Adopting and Evolving a Product Line Approach"*, Addison-Wesley, Harlow UK, 2000.

[Hofmeister et al. 2000] C. Hofmeister, R. Nord, D. Soni, *"Applied Software Architecture"*, Addison-Wesley, Reading MA., 2000.

[ISO 9126] *Software Qualities"*, ISO/IEC FDIS 9126-1:2000(E).

[Karlsson & Ryan 1997] J. Karlsson and K. Ryan, "A Cost-Value Approach for Prioritizing Requirements", in *IEEE Software* **14** (5):67–74, 1997.

[Karlsson et al. 1998] J. Karlsson, C. Wohlin and B. Regnell, "An Evaluation of Methods for Prioritizing Software Requirements", in *Information and Software Technology*, **39**(14-15):938-947, 1998.

[Saaty 1980] T. Saaty, *"The Analytic Hierarchy Process"*, McGraw-Hill, 1980.

[Saaty & Vargas 2001] T.L. Saaty, L.G. Vargas, *"Models, Methods, Concepts & Applications of the Analytic Hierarchy Process"*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2001.

[Shaw & Garlan, 1996] M. Shaw, D. Garlan, *"Software Architecture - Perspectives on an Emerging Discipline"*, Prentice Hall, Upper Saddle River NJ, 1996.

[Shepperd et al. 1999] M. Shepperd, S. Barker, M. Aylett, "The Analytic Hierarchy Process and almost Dataless Prediction", in *Project Control for Software Quality - Proceedings of ESCOM-SCOPE 99*, R.J. Kusters, A. Cowderoy, F.J. Heemstra, E.P.W.M. van Weenendaal (eds), Shaker Publishing BV, Maastricht the Netherlands, 1999.

[Svahnberg & Wohlin 2002a] M. Svahnberg, C. Wohlin, "An Investigation of a Method for Evaluating Software Architectures with Respect to Quality Attributes", Submitted, 2002.

[Svahnberg & Wohlin 2002b] M. Svahnberg, C. Wohlin, "Consensus Building when Comparing Software Architectures", in *Proceedings of the 4th International Conference on Product Focused Software Process Improvement* (PROFES 2002)*,* Lecture Notes in Computer Science (LNCS 2559), Springer Verlag, Berlin Germany, 2002.

[Svahnberg et al. 2002] M. Svahnberg, C. Wohlin, L. Lundberg, M. Mattsson, "A Method for Understanding Quality Attributes in Software Architecture Structures", in *Proceedings of the 14th International conference on Software Engineering and Knowledge Engineering* (SEKE 2002), ACM Press, New York NY, pp. 819-826.