M. Svahnberg and C. Wohlin, "Consensus Building when Comparing Software Architectures", Proceedings of the 4th International Conference on Product Focused Software Process Improvement (PROFES 2002), pp. 436-452, Springer LNCS Volume 2559/2002, Rovaniemi, Finland, December 2002.

Consensus Building when Comparing Software Architectures

Mikael Svahnberg, 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. When designing a software system it is beneficial to study and use architectural styles from literature, to ensure certain quality attributes. However, as the interpretation of literature may differ depending on the background and area of expertise of the person reading the literature, we suggest that structured discussions about different architecture candidates provides more valuable insight not only in the architectures themselves, but in peoples' opinions of the architectures' benefits and liabilities. In this paper, we propose a method to elicit the views of individuals concerning architecture candidates for a software system and pinpoint where discussions are needed to come to a consensus view of the architectures.

1 Introduction

When developing software, it is important to have an appropriate architecture for the system, or sub-systems comprising the full system. The choice of, or evolution into, an appropriate architecture is not only governed by functional requirements, but to a large extent by quality attributes [2][4][6].

However, knowing this, it is still a non-trivial task to discern between architecture candidates. There are usually more than one quality attribute involved in a system, and the knowledge of the benefits and drawbacks of different architecture structures with respect to different quality attributes is not yet an exact science. Decisions are often taken on intuition, relying on the experience of senior software developers.

Because of this it is important, we believe, to be able to compare software architecture structures based on quantified data. Likewise, it is important to be able to compare the strengths and weaknesses of a single software architecture structure based on quantified data. If this is not done, there will always be subjective judgements involved when selecting between architecture structures.

What is even more important is that everyone involved in designing a software architecture share the same view of what benefits and drawbacks different architecture structures have. To do this, it is important that the views of different persons are extracted in a quantified way that enables comparisons between the views, and a synthesis of the different views into a unified consensus view.

We propose that the understanding of architectures starts with eliciting the knowledge of individuals and that structured discussions should be held to reach a further understanding and learn from others during the process of building consensus around the benefits and liabilities of different architecture candidates.

It should be noted that we use the term "software system" rather loosely in this paper. We use it to mean any software entity, be it an entire product suite, a single product, a subsystem within a product, a software module, or a software component.

1.1 Scope and Goal of Paper

In this paper, we describe a process for capturing knowledge from individuals into a framework that enables analysis of and comparison between software architectures with respect to quality attributes. In the process of synthesizing this framework, the views of the individuals participating in the process are extracted and presented in a way that enables and facilitates discussion where the participants are in disagreement. The purpose of these discussions is to create a joint understanding of the benefits and liabilities of the different software architectures.

Throughout this paper we illustrate each step with data and experiences from conducting the step with the participation of colleagues, who have also participated in creating the initial data sets in a previously conducted experiment, described in further detail in [17].

We would like to stress the fact that even if we in this experiment used generic architecture structures and quality attributes one would, if using the method in a company, develop architecture candidates and elicit quality requirements for a particular system rather than using generic architectures and quality attributes. The method would thus operate on architectures that can be used in the particular system. Likewise, the quality attributes used would be elicited for the particular system, and would hence be expressed in terms pertinent to the system's problem domain.

Moreover, the focus of this paper is mainly on the process described and the discussion of this rather than on the example, which is mostly included to illustrate the different steps.

The remainder of this paper is organized as follows. In Section 1.2 we present an outline of the process proposed in this paper. In Section 2 we present how to create individual views, discuss these and combine them to a unified framework. In Section 3 we present how to evaluate this unified framework, and in Section 4 we present how to analyse it. Finally, the paper is concluded in Section 5.

1.2 Outline of Process

The process we describe in this paper consists of the following five steps (also illustrated in Figure 1), each of which we go through in further detail in this paper:

1. Create individual frameworks, as outlined in Section 2.1. The individual frameworks consists of two tables per participant, where one table describes the participant's ranking of the support for different quality attributes for each of the architecture structures, and the other ranks the architecture structures for each quality attribute.
2. Discuss the individual frameworks and decide upon a strategy for combining them into a unified consensus framework, which we go through in Section 2.2 in this paper.
3. Create the unified framework as presented in Section 2.3.
4. Evaluate Framework, as presented in Section 3.
5. Analyse the framework. This step is described in Section 4.

As presented by Johansson et al. [9], it is expected that stakeholders have different views of the importance of different quality attributes, and we also expect developers with different backgrounds to have different views of different architecture structures. The purpose of steps 1 and 2 of the process in this paper is hence to elicit the views of different stakeholders and use these as a basis for further discussions where the causes for the different views are investigated.

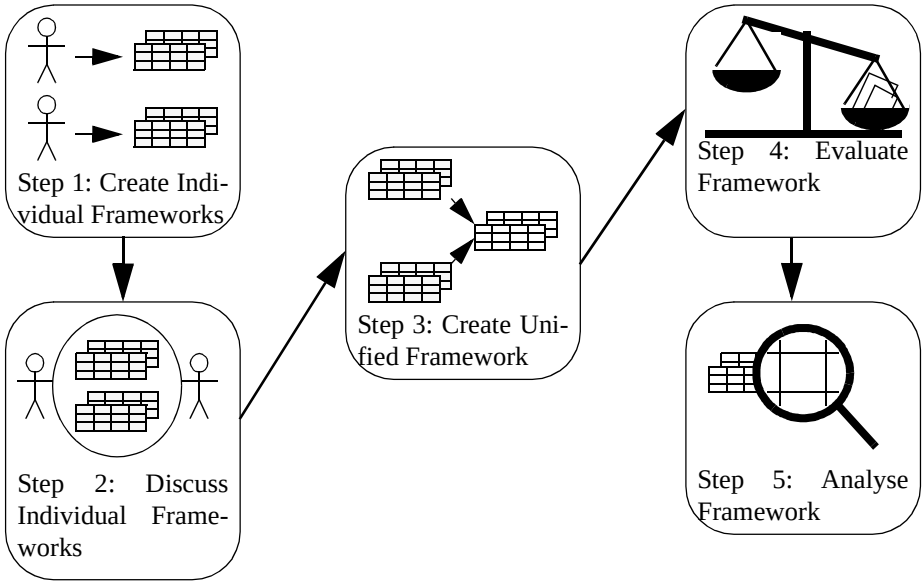

Figure 1. Illustration of Process

These discussions serve as one important input to the architecture design process and the development process, but it is also important to analyse the framework to understand the benefits of different architecture structures. This is not only useful if the architectures represent initial designs for a system, but is even more important if the purpose of the evaluation is to see whether the current architecture of an evolving system is still the most appropriate alternative given the (potentially also evolved) domain and quality requirements of the software system. By regularly re-evaluating the choice of software architecture phenomena such as software aging [13] may be, if not stopped, so at least slowed down. This analysis consists of steps 4 and 5, where the framework are first evaluated to measure the amount of unceirtanty it contains (step 4) and then each architecture candidate analysed in further detail compared to the other architecture candidates (step 5).

2 Creating the Framework

The framework we intend to create consists of two tables, which we refer to as the *FQA (Framework for Quality Attributes)* and the *FAS (Framework for Architecture Structures)*. These two tables consist of a set of vectors, normalized so that the values in each vector sum up to 1, and the tables describe the architecture structures with respect to quality attributes in two ways: the FQA describes a ranking of architecture structures with respect to a certain quality attribute, and the FAS describe the ranking of different quality attributes within a particular architecture structure.

We create these two tables by first acquiring the individual views of a number of participants, as described in Section 2.1 and then discussing these views in a meeting, which we describe in Section 2.2. The last step is to combine all individual views into a unified framework, as described in Section 2.3.

However, before the framework can be created, it must be known which quality aspects are relevant to consider, and a set of architecture structures must be developed for the system to design. We do not describe this further, as it is done using traditional requirements engineering (e.g. [5][12]) and architecture design methods (e.g. [4][6][8]). The input for the method is thus highly context dependent, as the architecture structures and the quality attributes are designed and elicited for a particular software system, in a particular domain and for a particular software company with a certain software development culture.

The elicited relevant quality aspects and the architecture structure candidates are used as input to the first step of the process, as described below.

2.1 Creation of Individual Frameworks

The first step after the architecture structures are created is to understand how different individuals, potentially with different backgrounds, perceive the strengths and weaknesses of the architecture structures with respect to a set of quality attributes.

One can choose whether to elicit the views, in terms of the FAS and FQA, of each individual or whether to obtain a collective framework in a group meeting. If the participants create the framework collectively, there is a risk that opinions are suppressed, wherefore we suggest that it is better if each participant create an individual framework that becomes the basis for discussions, e.g. as presented in Section 2.2. Each of these individual frameworks should then describe the strengths and weaknesses of the architecture structures in a comparable way, so that differences between participants can be identified and discussed.

However, to explain in comparable terms what the strengths and weaknesses of a particular architecture structure are, is not a trivial task. To facilitate this, we propose the use of methods available from the management science literature, for example in Anderson et al. [1]. The methods are often denoted multi-criteria decision processes.

One such method is the Analytic Hierarchy Process (AHP for short), which was originally proposed by Saaty [14] (and also described in a later publication [15]). This approach has been applied in software engineering by other researchers addressing, for example, requirements engineering [10] and project estimation [16]. The Analytic Hierarchy Process can be used to prioritize different items or aspects. The result is a priority vector with relative weights on the different items or aspects being prioritized.

In order to create the individual frameworks, what needs to be done is to complete two AHP questionnaires, with questions pertaining to the following two issues:

- A comparison of different quality attributes for each software architecture structure.
- A comparison of different software architecture structures for each software quality attribute.

This will then result in two tables per participant, related to the FAS and the FQA as earlier described.

Illustration of Creating Individual Frameworks. In a previous study [17], we present an experiment where we conduct the type of AHP ranking as mentioned above, i.e. we describe and apply a method for assessing the support different architectural structures have for different quality attributes, and also which architectural structures best fulfil certain quality attributes.

The outcome of this experiment is a series of vectors for each participant in the study. In our study each of the eight participant produced six vectors ranking architectural structures for each quality attribute and five vectors ranking the quality attribute

potential within each architecture structure. These vectors are grouped into two tables per participant, corresponding to the FAS and the FQA as described above.

The quality attributes used were those from the ISO 9126 standard [7], namely: Efficiency, Functionality, Usability, Reliability, Maintainability and Portability, and the architecture structures used were a selection from Buschmann et al. [3], namely: Microkernel, Blackboard, Layered, Model-View-Controller and Pipes and Filters. It should however be noted that the method is not bound to these attributes and structures in particular. Any other set would have worked just as well.

The individual frameworks can be studied separately, but the real use comes if they can be combined into a single, comparable view of the architecture structures and quality attributes, e.g. as described in the next sections.

2.2 Discussing the Individual Frameworks

The purpose of discussing and comparing the individual frameworks is to create a further understanding of where the software engineers disagree in their judgements of the architecture candidates, and to elicit the reasons why this disagreement occurs. We expect to find disagreements, as it is rare that all software engineers have the exact same background and experience, and these differences will be manifested in the individual frameworks created in the previous step.

In order to identify the discrepancies that are most relevant to discuss, we propose to use the sum of the squared distance to the mean value, described by the following formulae: $\sum_{i=1}^{N} (x_i - \bar{x})^2$, where N is the number of participants. This formulae is applied over all participants for each vector in the FAS and FQA (in effect, for each quality attribute and each architecture structure), and hence produce a value for each vector that describes the amount of disagreement between the participants. After this, a suitable threshold value is selected to discern which of the vectors are worthy of further examination.

Although it is up to the user of our method to set a suitable threshold value and it is depending on the number of discussion points one wish to identify, we suggest that the threshold value is set to the 75th percentile, thus pinpointing the upper 25% of the data set. However, this also needs to be augmented by visually inspecting graphs of the individual answers and identifying places where there are interesting outliers even though the spread of the answers do not exceed the threshold value.

During a meeting, each of the identified data points are discussed, and participants with a differing opinion from the rest get a chance to explain why their values differ.

Embracing Disagreement. That different persons have different backgrounds is not an uncommon situation, neither in academia nor in industry. Thus, any formed group will consist of persons with different backgrounds, which is partly what makes a group successful. As all group members form their interpretations of the situation at hand based on their background, one cannot expect all participants to have the same interpretation. We believe that the key to success is to acknowledge this and to find ways to cope with the differing interpretations.

If participants disagree on the meaning of a certain quality attribute, or of a certain architecture structure, this is a disagreement that would manifest itself later during the development process and, in a worst case, be the source of flaws in the delivered product.

The major contribution of the meeting presented in this section is that the participants get to present their rationale, and this creates a better joint understanding of how to interpret the quality attributes and architecture structures.

Another goal of the discussions is that the individual vectors should be combined into vectors that everyone can agree upon. There are several ways to combine the vectors, e.g.:

- Use the mean value.
- Remove outliers and use the mean value.
- Use the median value.
- Let the participants, with the gained knowledge from the discussion, re-do the AHP questionnaire for the vector in question and hold a second meeting to discuss the new vector.
- Let the participants jointly complete an AHP questionnaire for the vector in question.

Which method to use can be decided during the meeting for every vector, but we suggest that in most cases using the median value is sufficient. It is less time-consuming than the other choices, while still giving a more accurate image than just using the mean value. The mean value would be unduly influenced by extreme values, whereas the median value indicates where the bulk of the participants are located without biasing towards outliers.

Illustration of a Consensus Discussion Meeting. With the goal of creating a unified view of the eight different opinions (stemming from the eight different participants in the previous study [17]), we conducted a follow-up meeting. During this meeting, the 11 calculated vectors (one vector for each architecture structure and quality attribute used: 5 architecture structures + 6 quality attributes) per participant based on the AHP study was presented and then discussed from the perspective of a smaller set of data points which was deemed worthy of further examination. These data points include those where there is a large spread among the answers of the participants, and those where the participants' opinions form two, or in some cases three distinct groups.

As a guideline for finding these data points, we used the sum over all participants of the squared distance to the mean value, with a threshold value of 0.10, which roughly corresponds to the 70th percentile. Hence, any data point where the sum over all participants of the squared distance to the mean value was larger than 0.10 was deemed interesting enough to warrant further discussion.

Using this simple technique, we identified 20 data points out of 60 (Five architecture structure vectors with 6 values each, and six quality attribute vectors with 5 values each equals 60 data points per participant, and we are looking across all of the participants) that warranted discussion. Of these, 6 data points were only marginally over the threshold value and were not discussed in as great a detail. Even though the set threshold value roughly corresponds to the 70th percentile, we thus only held detailed discussions about the data points above the 75th percentile.

In addition to the data points identified by the threshold value described above we also noticed, while studying graphs of the data sets, that in some cases one or two participants disagreed largely with the rest of the group. We included these data points as discussion points as well, as it is important that all arguments are heard, and the disagreeing person or persons may have very compelling reasons for disagreeing with the rest of the participants.

As stated, the intention of the discussion meeting is to find out the specific reasons for why the participants may have differing opinions in the identified data points. In our

case, it soon became apparent that all disagreements could be put down to the same factor, namely that the interpretation and application of architecture structures are dependent on the background of the participants. This led people with different backgrounds from different disciplines to interpret the architecture structures differently. As the architecture structures in themselves are rather abstract, many of the participants envisioned a typical system in which the architecture is used, to put a context to the question. These envisioned systems differed depending on the background of the participants.

If the framework were created and the discussions were held in an industry case this would, however, not be an issue, as the context in that case is given by the software system in focus, and the architecture structures and quality attributes directly relatable to this system. Moreover, this difference in the systems envisioned is of minor importance to this paper, as the focus is on the presented process for eliciting and analysing peoples' opinions of different architecture candidates, and to study the problems surrounding the creation of a consensus view of the strengths and weaknesses of the different alternatives.

2.3 A Unified Framework

After conducting the meeting described above, where the disagreements are discussed, a unified Framework for Architecture Structures (FAS) and a unified Framework for Quality Attributes (FQA) is constructed of the participants' views using the method decided upon to unite the individual views. Most often, the median value will be sufficient, unless arguments are brought forward to use another method (e.g. the ones mentioned in the previous section) for uniting the individual frameworks.

By using the median value, these unified frameworks are no longer normalized as the individual tables were, i.e. the columns in the FAS and the rows in the FQA no longer sum up to 1. Because of this, a step is added where the data is re-normalized so that the columns in the FAS and the rows in the FQA sum up to 1.

Illustration of Unified Framework. The FAS and FQA constructed from our study after the consensus discussion meeting are presented in Table 1 and Table 2.

The FAS (Table 1) presents the ranking of quality attributes for each architecture structure. This table should be read column-wise. For example, it ranks microkernel as being best at portability (0.309), followed by maintainability (0.183), efficiency (0.161), reliability (0.122), functionality (0.119) and usability (0.106), in that order. Moreover, the figures indicate that for example microkernel is almost twice as good at portability as it is at efficiency (the value for microkernel is 0.309 compared to the value for efficiency which is 0.161).

The FQA (Table 2) presents the ranking of architecture structures for each quality attribute, and should be read row-wise. For example, the FQA ranks pipes and filters as the best choice for efficiency (0.360), followed by microkernel (0.264), blackboard (0.175), model-view-controller (0.113) and layered (0.0868), in that order. As with the FAS, the figures indicate how much better a choice for example pipes and filters is compared to the other architecture structures. It is, for example, twice as good a choice as blackboard (with a value of 0.360 compared to the 0.175 that blackboard scores).

3 Evaluation of Unified Framework

Previously we discussed the importance of embracing disagreement. This is not only done by venting peoples opinion in a meeting, as earlier described. For each value in the FAS and FQA, a value can be added to indicate the amount of disagreement between

Table 1. Framework for Architecture Structures (FAS)

	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 2. Framework for Quality Attributes (FQA)

	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

the participants. Such disagreement indicators can be used to judge the accuracy of decisions or statements based on data from the framework.

Disagreement indicators can be constructed in a number of ways, but we suggest that the same measure as earlier is used, i.e. the squared distance to the mean, and count the number of participants with a larger value than a certain threshold.

As before, the idea is to set the threshold such that it identifies where the participants actually are in disagreement, which means that if the threshold is too high too much disagreement is allowed, and if it is too low there is too little tolerance for variations in the answers.

However, it is not feasible to set the threshold value to identify a particular percentile as we did to identify data points that warrants discussion. Instead, we need a value that correctly depicts the amount of disagreement found and not a value that identifies a particular group of data points. To this end, we recommend that points where the squared distance to the mean is larger than two standard deviations (of all the squared distances to the mean) are deemed to be in disagreement with the rest of the participants. As before, what value to use as a threshold value is up to the user of the method, but we find that two standard deviations give a fair picture of the amount of disagreement.

This measure of disagreement is only one in a series of uncertainty indicators. For every step of the way, we have indicators of uncertainty, and these should be considered so that, if the uncertainty becomes too large, it should be possible to backtrack and redo steps to get more certainty in the data sets and hence in the accuracy and usability of the framework:

The uncertainty indicators available hitherto are:

1. Individual consistency ratio for each of the produced vectors. If a method such as AHP [14][15] is used, this is obtained as part of the results from the method, otherwise these may need to be calculated separately.
2. Differences between individuals, as discussed in Section 2.2 and using the measure introduced there.
3. Differences between the unified FAS and FQA. In [18] we describe a way to measure and compensate for these differences. Briefly, we compensate for inconsistencies between the FAS and FQA using one of the frameworks to improve the quality of the other, which is then used in the subsequent steps of the method.

In every step of the way, the goal has been to quantify the knowledge about architecture structures, while still retaining a qualitative rationale. Every step of the way helps in removing ambiguity and increasing the clarity and the understanding of the architecture structures. This will, in a development process, ensure that architecture design decisions can be taken with more certainty.

The uncertainty indicators on all levels and during all steps of the creating of the framework can be used to ascertain that the uncertainty, and hence the risk involved, is reasonably low, but also to identify factors upon which people have different opinions and where further discussions are needed to avoid problems further on in the development process.

Illustration of Disagreement Measure. In our example, the standard deviation of all squared distances to the mean value is 0.0166, and hence the threshold value is set to the double, i.e. 0.0332. By confirming against a plotting of all the participants, we are able to determine that this threshold value gives a fair representation of where participants disagree with the majority.

Counting the occurrences where the participants in the study diverge more than this threshold number, we find that there are 43 places where participants disagree, out of the total 480 data points (6 vectors with 5 values plus 5 vectors with 6 values, and all this times 8 participants). The persons in disagreement are distributed over the different vectors as shown in Table 3 and Table 4.

In these tables, we see for example in Table 3 that for Microkernel one person had a different opinion to that of the majority regarding its efficiency value, one person regarding its usability value and as many as four persons disagreed on Microkernel's abilities regarding portability. Studying a graph with all the participants' individual frameworks, it becomes clear that the participants form two distinct groups with respect to this issue (although the points identified by the disagreement measure come from both of these two groups).

Moreover, we see that in general the architecture structures Microkernel and Blackboard contribute with more than half of the disagreement issues, which indicates that for these two architecture structures much discussion is needed in order to fully understand them, and the consequences of using them in a software system.

In Table 3 and Table 4, we see that there are a total of eight places where two or more participants disagree with the majority. While these places certainly need further discussions to elicit the reasons for the disagreement, we can also conclude that the unified framework seems to be constructed by persons who are mostly in agreement, and the framework can thus be used with reasonable accuracy.

Table 3. Disagreement in FAS

	Microkernel	Blackboard	Layered	Model-View-Controller	Pipes and Filters
Efficiency	1	1			1
Functionality		1			
Usability	1				
Reliability				1	
Maintainability		1	1		1
Portability	4	1			1

Table 4. Disagreement in FQA

	Microkernel	Blackboard	Layered	Model-View-Controller	Pipes and Filters
Efficiency	2			2	
Functionality	1	4		1	
Usability	1	1		2	
Reliability	1		2	1	1
Maintainability		1	4	1	1
Portability	2				

4 Analysis of Framework

In this section, we describe the logical next step after the framework is evaluated for consistency, which is to analyse the framework internally, i.e. to discern how the different architecture structures relate to each other and how each of the architecture structures support different quality attributes.

This is important in order to really understand the relations between the architecture structures and the quality attributes. Moreover, to analyse the framework instead of simply using it creates a learning effect, in that by understanding the qualities of one software architecture, it may be easier to understand the qualities of the next architecture, i.e. the next time software architectures are designed and when evolving the software architectures, the designers will have an increased understanding from the start of the strengths and weaknesses of different design alternatives.

Furthermore, if the purpose of creating the framework is to re-evaluate the architecture of an existing software product, it becomes even more vital to analyse the architecture alternatives in the created framework to understand for which quality attributes there is an improvement potential in the current software architecture of the system.

The analysis is based on the two tables, i.e. the FAS and the FQA. We have attempted several ways to integrate these two tables into a single table, but have come to the conclusion that it is better to keep the two tables separate.

There are two dimensions to the analysis: (a) a comparison between different architecture structures, for which the FQA is used, and (b) a comparison of the software qual-

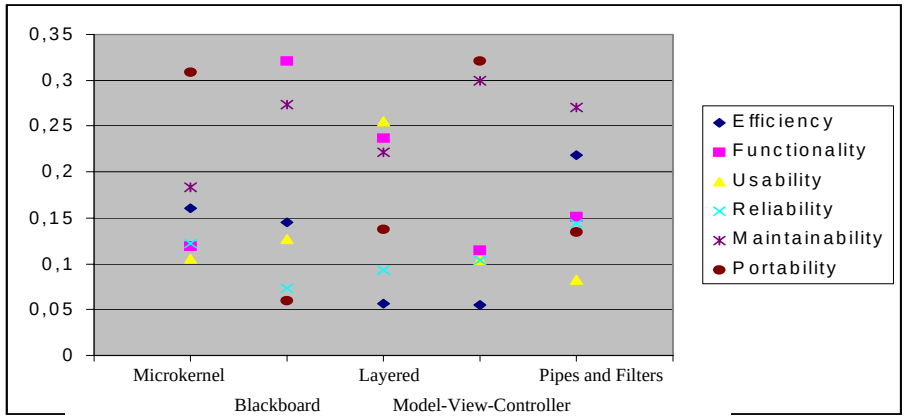

Figure 2. Plotting of FAS

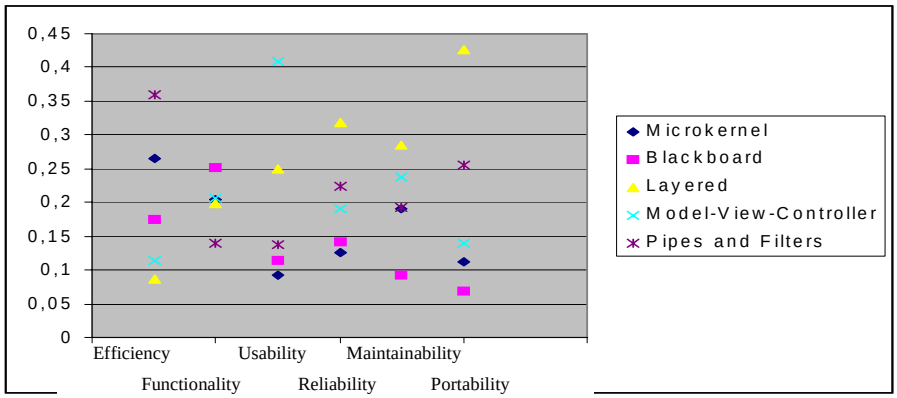

Figure 3. Plotting of FQA

ities within a particular architecture structure, for which the FAS is used. As before, the FQA is read row-wise, and the FAS is read column-wise.

Rather than studying the mere numbers, which can be difficult to compare, we suggest that the data is plotted into graphs. Examples of such graphs, based on the FAS in Table 1 and the FQA in Table 2, can be found in Figure 2 and in Figure 3.

The graph in Figure 2 is read column-wise, i.e. one column in the FAS correspond to one column, or rather a vertical line, of dots. Figure 3 corresponds to the FQA, which is read row-wise, but as there is no meaningful interpretation of comparing the quality attributes with each other in this table, we choose to plot this table so that its graph is also read column-wise.

4.1 Analysis of Architecture Structures (FAS)

This part of the analysis is concerned with understanding the qualities of each architecture structure. To this end, we use the FAS, and a plot of it, as the example in Figure 2.

When studying the FAS, it becomes apparent that an architecture structure can be of two different kinds, depending on how the quality attributes are spread. If the architecture ranks a few quality attributes highly compared to the others, this implies that the architecture is specialized for these purposes. On the other hand, if all of the quality attributes are ranked closely together, the architecture is generalized, and can better suit any mixture of desired quality attributes.

Another thing to look for is whether some quality attributes are always, or in most cases, ranked high or low. For example, if a quality attribute is ranked high for all architecture structures, there is no further need to study this attribute as one can be certain to fulfil it in any case. If it is ranked low, this indicates that new architecture structures may be needed that focus on this quality attribute.

The purpose of this analysis is to understand the strengths and weaknesses of each architecture structure, which may help the next time an architecture is to be designed. As stated, it may also help in pointing out the need for new architecture structures that either favours a particular quality attribute, or is equally good at a number of quality attributes.

Illustration. In Table 1 (and in Figure 2) all of the architecture structures, except for possibly the pipes and filters structure, are specialized for certain purposes. For example, microkernel is specialized for portability. Portability scores a value of 0.309 compared to the second largest, which is maintainability with 0.183. The relative distance between portability and maintainability is thus 0.126, whereas the relative distance between maintainability and the quality attribute with the lowest score is 0.0770, which is considerably smaller than between portability and maintainability. Another example of a specialized architecture structure is model-view-controller which is specialized on portability and maintainability, with values of 0.320 and 0.300, respectively. The distance to the third best quality attribute is more than 0.185, as compared to the distance between the third best and the last quality attribute, which is 0.0593.

Another thing that the FAS from our experiment seems to indicate is that there are some quality attributes that no architecture structure is really good at. For example, all architecture structures except for pipes and filters have a fairly low value on efficiency. What this implies is that there is room for specialized architecture structures with emphasis on this quality attribute.

Similar situations can be found with usability, where only the layered architecture structure ranks it highly, and reliability, which no architecture structure in our study seems to focus on, or is capable of.

Since no architecture structures in our study focus on reliability, one can ask the question whether reliability is at all a quality attribute that affects the software architecture or whether it is an attribute that mainly manifests itself in the software development process. This, in turn, may be depending on the interpretation of reliability.

Maintainability, on the other hand, seems to be a quality attribute that most, if not all, architecture structures seem to focus on. This manifests itself in the relatively high values for maintainability compared to the other quality attributes for all architecture structures.

4.2 Analysis of Ranking per Quality Attribute (FQA)

The other part of the analysis is to compare architecture structures with each other. To this end we use the FQA, and a plot of this, as is exemplified in Figure 3.

Patterns one can discern here are whether a particular architecture structure always gets better values than another, which of course would mean that this architecture struc-

ture is always to prefer over another. One can also see if there are certain quality attributes that favours the selection of a particular architecture structure. This may, in time, create an increased understanding of what traits of a software architecture it is that benefits a particular quality attribute.

Illustration. In the FQA in Table 2 (and the corresponding plot in Figure 3), the layered architecture is considerably better at most quality attributes than the other architecture structures, except for efficiency and functionality (and usability, where model-view-controller is extremely better than the rest of the architectures). Likewise, we see that blackboard is in general a bad choice, except when functionality is a desired quality. If a situation like this occurs when comparing architecture structure candidates for a system, this would indicate that the evaluation can be aborted and the high-ranking architecture (in our example the layered architecture structure) can be chosen directly, unless efficiency is a desired quality.

5 Conclusions

In this paper we present a way to build consensus around the benefits and liabilities of software architecture structures through the process of creating a unified, quantified and comparable view of architecture structures with respect to quality attributes.

Instead of just reading about the architecture structures in a book, we believe that the process of expressing ones own knowledge and experience of different architecture structures in a structured way creates a further understanding of how the architecture structures will work in the specific situation.

When this is compared to other peoples opinions as well, this creates a learning effect and allows differences of opinions to be identified and discussed to form a consensus before the development process continues.

Without this consensus, it is our belief that the differences of opinion will appear later during the development, and take the form of inconsistencies in the developed system, and an increased development time.

Unlike related literature (e.g. [3][4]), which only present benefits and liabilities by means of logical reasoning, a framework created using the process in this paper provides relative measures of the level of support for different quality attributes, thus enabling measurement of the importance or severity of different traits. Furthermore, we also provide a way to compare these traits over different architecture structures. This complements the work of e.g. Buschmann et al. [3] and Bosch [4] by providing an opportunity to quantitatively analyse architecture structures, and thus create a further insight into how these architecture structures work.

Moreover, the framework can be constructed for any set of architecture structures and quality attributes, which means that companies can perform their own analysis for architecture structures and quality attributes related to their business and the domain of their systems, and is hence not bound to the selection of architecture structures and the descriptions of these that can be found in mainstream literature.

The focus of the paper is on the different steps that assist in the process of building consensus among the participants while ensuring that all, or at least many, relevant aspects are covered before more time and effort is spent on further developing the software system at hand.

We illustrate these steps by reporting from a case study conducted according to the steps in this paper. Hence, we would again like to stress that the framework we present in the illustrations is only one example of using the proposed process. The created framework and the steps to discuss, evaluate and analyse it can be used on any set of

architecture structures and quality attributes. Which sets to use is primarily determined by the context and the domain in which the process is being applied.

Moreover, we would also like to stress that albeit the framework is constructed by capturing the *perception* of architecture structures and quality attributes from professionals, it is our belief that the perception professionals have about architecture structures are also represented as actual qualities of the architecture structures themselves. Nevertheless the framework is only indirectly based on the actual qualities of the architecture structures.

The process for consensus building in this paper has the following benefits:

- It can be used to create a better understanding of different architecture structures.
- It can be used to kindle a dialogue between software developers to iron out and understand discrepancies in interpretations of architecture structures.
- It can be used to identify the need for architecture structures specialized on certain quality attributes.
- It can be used as a sub-step in methods for comparing different architecture structures when selecting which architecture to use in a system to design, as in [18].
- It can be used to evaluate architectures against a "baseline" of common architecture structures.
- It can be used as a learning tool to allow software developers to share their experiences with each other in a structured way.
- It can be used to confirm or confute "myths" regarding software architectures. For example if all architecture structures rank performance and maintainability highly, this indicates that it is at least possible to create architecture structures where these are not in conflict, thus refuting the myth that this, in general, is not possible.

To summarize, the contribution of this paper is that we present a process for creating a data set around which discussions can be held to find out if and why there are disagreements in a group of software developers. Such a discussion is, we believe, vital to create a joint understanding of the architecture candidates for a system to design. Our advice is to acknowledge that there will be disagreement, and to use this disagreement to power discussions to create a better understanding of the architecture structures and quality attributes involved.

References

[1] 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.

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

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

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

[5] L. Chung, B.A. Nixon, E. Yu, J. Mylopoluos, *"Non-Functional Requirements in Software Engineering"*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2000.

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

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

[8] I. Jacobson, G. Booch, J. Rumbaugh, *"The Unified Software Development Process"*, Addison-Wesley, Reading MA, 1999.

[9] E. Johansson, M. Höst, A. Wesslén, L. Bratthall, “The Importance of Quality Requirements in Software Platform Development - A Survey”, in *Proceedings of HICSS-34*, Maui Hawaii, January 2001.

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

[11] 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.

[12] G. Kotonya, I. Sommerville, *“Requirements Engineering”*, John Wiley & Sons, Chichester UK, 1998.

[13] D.L. Parnas, “Software Aging”, in *Proceedings of the 16th International Conference on Software Engineering*, IEEE Computer Society Press, Los Alamitos CA, pp. 279-287, 1994.

[14] T. Saaty, *“The Analytic Hierarchy Process”*, McGraw-Hill, 1980.

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

[16] 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.

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

[18] 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.