M. Svahnberg and C. Wohlin, "An Investigation of a Method for Identifying a Software Architecture Candidate with Respect to Quality Attributes", Empirical Software Engineering: An International Journal, Vol. 10, No. 2, pp. 149-181, 2005.

An Investigation of a Method for Identifying a Software Architecture Candidate with respect to Quality Attributes

Mikael Svahnberg
Mikael.Svahnberg@bth.se

Claes Wohlin
Claes.Wohlin@bth.se

Department of Software Engineering and Computer Science
Blekinge Institute of Technology, PO Box 520, S-372 25 Ronneby, SWEDEN
Phone: +46 457 385000

ABSTRACT

To sustain the qualities of a software system during evolution, and to adapt the quality attributes as the requirements evolve, it is necessary to have a clear software architecture that is understood by all developers and to which all changes to the system adheres. This software architecture can be created beforehand, but must also be updated to reflect changes in the domain, and hence the requirements of the software. The choice of which software architecture to use is typically based on informal decisions. There exist, to the best of our knowledge, little factual knowledge of which quality attributes are supported or obstructed by different architecture approaches. In this paper we present an empirical study of a method that enables quantification of the perceived support different software architectures give for different quality attributes. This in turn enables an informed decision of which architecture candidate best fit the mixture of quality attributes required by a system being designed.

Keywords

Software Architectures, Quality Attributes, Analytic Hierarchy Process

1. INTRODUCTION

In (Parnas 1994) Parnas describes the phenomenon of software aging. He ascribes this to two causes: (1) the domain changes around the software and (2) changes to the system are introduced in a careless manner, which degrades the system. Part of the solution to both of these problems may be found in having and maintaining a clear and updated software architecture for a software system. To keep this architecture consistent with a changing domain, it needs to be regularly re-evaluated. By doing this on a regular basis, we believe that the first form of aging can be, if not hindered, so at least relieved.

An appropriate architecture is not only governed by functional requirements, but to a large extent by quality attributes (Bass et al. 1998; Bosch 2000; Hofmeister et al. 2000). However, knowing this it is still a non-trivial task to create an appropriate architecture. There are usually more than one quality attribute involved in a system, and the knowledge of the benefits and drawbacks of different software architecture approaches 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.

This imposes a problem because, as shown by (Johansson et al. 2001), different stakeholders tend to have different views of the importance of various quality requirements for a system, and the differing experiences of the software developers may also lead to a different interpretation of the strengths and weaknesses of architecture structures.

A structured decision support method facilitates in this situation because it enables us to identify where stakeholders and developers have differing opinions and discuss these at an early stage, before these differences of opinion may cause problems during the development process. Moreover, a decision support method that facilitates in structuring the knowledge of different architectures may increase the confidence in the decisions taken.

This paper presents an investigation of a method that structures the knowledge and previous experiences of subjects when assessing the support given for different quality attributes in a set of software architecture candidates for a system being built. The goal of this method is to enable software designers to select, among a set of architecture candidates, the architecture candidate which is most suitable for a particular system. This is determined by assessing which architec-

ture candidate has the best potential for fulfilling the blend of quality requirements for the system in question. The method also pinpoints where the subjects do not share a common view of the benefits and liabilities of the architecture candidates so that focussed discussions can be held.

The investigation illustrates how different individuals understand and judge the support of quality attributes in a comparison between software architectures. The investigated method provides one important input to decision-makers when selecting a suitable system architecture, together with other considerations. For example, these other considerations involve all aspects of software development that are not measurable on the software architecture or indeed the developed software system, such as the development organisation.

We do not put any restrictions of the size or scope of the software architectures or the software systems to which the method is applied. This can range from parts of a system to subsystems or the entire system. We perceive a software architecture to consist of the overall structure of a software system, e.g. using the triplet elements, form and rationale as described by (Perry & Wolf 1992), or the tuple components and connectors that (Shaw & Garlan 1996) and (Bass et al. 1998) discuss. However, for the sake of the method evaluated in this paper any definition of software architecture is usable, as long as all involved participants agree that the used definition of software architecture provides sufficient detail to enable a comparison of different architecture candidates.

Similarly, we do not put any constraints on the complexity of the quality attributes. However, in our experience it is beneficial if all quality attributes are on the same level of granularity, as this facilitates comparison between the quality attributes. Otherwise there is a risk that a more complex quality attribute may dominate the simpler quality attributes. Moreover it may be easier if the specific quality attributes are grouped into categories to facilitate the prioritization process. The reason why this is easier is simply that the number of inputs to the creation of the framework decreases, which reduces the number of comparisons that need to be made.

The paper is organised as follows. Related work is presented in Section 2. Section 3 discusses the problem addressed and the research questions related to the problem. In Section 4, the study design is presented to set the foundation for the results obtained. The operation of the study is described in Section 5. The results of the investigation and an interpretation of the results are provided in Section 6. Finally, a summary and some conclusions are presented in Section 7. In the remainder of this section we describe the scope of the study, an outline of the studied architecture evaluation method, and a description of the context in which the architecture evaluation method is intended to be used, since this differs slightly from the scope and the way we perform this study.

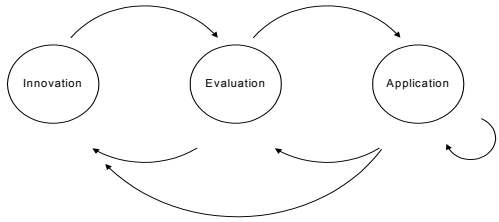

Figure 1. Technology Transfer Process

1.1 Scope of Study

This paper is part of a technology transfer process consisting of *innovation*, *evaluation* and *application*. This process is illustrated in Figure 1. This process is a subset of the technology transfer processes described in (Wohlin et al. 1996) and (Linkman & Rombach 1997).

In this process the first step consists of innovation, where an idea is born and developed to a “product”. The next step is to test, or evaluate, the method. This is preferably done in a lab environment, where more parameters can be controlled and where the cost for making mistakes is relatively low. Based on what is learnt here one can either move back to the innovation phase to further understand the phenomenon and evolve the idea or move on to the application phase where the idea or hypothesis is tested in real life settings. The application phase is thus the last step. In this step the idea has been tested in a lab environment and is proven to be useful. Moreover, many of the teething problems that any innovation suffers from have been ironed out. In other words, the idea is developed to such a level that it is ready to be tested in a live environment, where not all parameters can be controlled. It is possible to go back to the previous steps also from the application step, if issues are found that need to be further investigated or tested. Otherwise, the now fully developed product remains in the application phase,

where it is used in live settings. At this stage it may also be economically viable to develop tool support for the idea.

The contribution of this paper is in the evaluation phase, where we test certain aspects of an architecture evaluation method in a lab environment (in our case, an academic setting). As described in Section 1.3, the intended usage of the investigated method is ultimately in an industrial context where there is a known target system and the architecture candidates and quality attributes used in the method are specific for this known target system. However, before testing the method in an industry setting, it is important to study whether people will be in reasonable agreement about well known architecture candidates with well known benefits and liabilities. If it is difficult to study agreements and pinpoint causes for disagreements in this generic setting, this would be grounds for revising the method further before applying it in an industrial setting. Hence, we focus in this study on an academic setting using generic architecture styles (i.e. templates on which to base a concrete software architecture) with known benefits and liabilities, and generic quality attributes, rather than specific architecture candidates and quality requirements for a particular software system.

1.2 Method Outline

In (Svahnberg et al. 2002) we present a method for pinpointing a suitable software architecture for a system among a number of architecture candidates. The method provides a structured and formalised way to identify the most suitable software architecture candidate. This involves creating three different vectors or sets of vectors containing:

- A prioritized list of quality attributes for the target system.
- A comparison of the different software architecture candidates for each software quality attribute.
- A comparison of the different quality attributes for each software architecture candidate.

This method consists of the following steps:

Prepare Data. In this step we create and describe the software architecture candidates and the quality attributes to use in the study.

Create Individual Frameworks. Each of the participant in the study completes a questionnaire to create an individual version of the two aforementioned sets of vectors (i.e. two sets of vectors for each participant in the study are created). In addition, each participant in the study also creates a prioritized list of the quality attributes. We focus the study in this paper on this particular step.

Analyse Individual Frameworks. The individual frameworks are then analysed for internal consistency and are then compared to the other individual frameworks. The outcome of this step is to identify where the participants are in agreement and where there are differing opinions regarding the strengths and weaknesses of the different architecture candidates.

Discuss Individual Frameworks. In this step the points where there are disagreement, as identified in the analysis of the individual frameworks, are discussed in a meeting. These discussions are used to create a joint understanding of the benefits and liabilities of the software architecture candidates. The outcome of these discussions is to have a joint understanding of the architecture candidates, the quality attributes, and where more work is needed to really understand the impact of the quality attributes.

Suggest Software Architecture. The different individual results are combined and used to calculate which architecture candidate best match the quality requirements (i.e. the blend of quality attributes) of the system. However, we do not suggest that this recommendation should be used without reflection. Instead, we suggest that the recommendation is used to spark discussions during the discussion meeting described above.

1.3 Context of Method

The intention of the method investigated in this paper is to act as one input to decision makers, together with other inputs. The method focus on the perceived potential different architecture candidates have to support different quality attributes. The keywords here are thus *perceived* and *potential*.

The method measures the perceptions different persons have of what the benefits and liabilities are of different architecture candidates. These perceptions are coloured by the participants' previous experiences, and any bias they may have towards a particular architecture candidate, e.g. stemming from having built systems using a similar architecture previously. We argue that this is not a problem, since this experience and bias is also likely to facilitate development of a system using the favoured architecture candidate com-

pared to using a new and previously untested type of architecture.

A software architecture can, at the early development stages, only express a potential for achieving particular quality goals. There are many other aspects of software development, e.g. lower level design and implementation decisions and even the organisation of the development department that may cause a system's quality levels to deviate from this potential. However, this should not hinder at least making an effort to ascertain a software architecture that has a potential for supporting a particular blend of quality requirements at an early stage, to avoid forcing the system into a mould that is unsuitable for the required blend of quality attributes.

We would like to stress again that although we in this evaluative study use generic architecture styles and generic quality attributes without a specific target system, this is not the intended application of the method. As stated in Section 1.1, the purpose of this study is to establish some initial results and determine whether it is feasible to continue to an application of the method in its real, industrial context. In the intended industrial setting the method would take as input a known target system, a number of concrete architecture suggestions for this target system, and a set of quality attributes that are relevant for this particular target system. In this context, the main purpose of the method is to identify those aspects where people have different opinions so that further investigations can be made before problems arise from these different opinions during development, and to act as a decision support when selecting a software architecture for the software system.

2. RELATED WORK

Architecture evaluations can be separated into e*arly architecture evaluations* and *late architecture evaluations* (Lindvall et al. 2003), and different methods are more or less suitable for either of these two types.

Late architecture evaluation is conducted during later stages of the software development process when there is a software system or at least a detailed design available on which more concrete metrics can be collected. (Lindvall et al. 2003) is an example of this.

Early architecture evaluation, on the other hand, is concerned with deciding at an early stage during the software development process what qualities a software architecture have a potential for exhibiting. The intention of an early architecture evaluation is thus to act as a decision support when determining whether to continue development with a particular software architecture candidate, whether a developed software architecture candidate can be or need to be further improved before development commences, or whether a new software architecture candidate needs to be developed. This is also the intention of the method evaluated in this paper.

Early architecture evaluations are commonly based on the experiences of the software developers and logical reasoning, as there are usually no tangible artifacts on which to e.g. perform simulations or collecting metrics. Oftentimes, this is aided by first specifying, categorising and prioritizing scenarios. These scenarios then enables the evaluators to focus on one issue at a time.

Examples of evaluation methods focused on early evaluation and using scenarios are the Software Architecture Analysis Method (SAAM) (Bass et al. 1998) that is solely based on the use of scenarios, its successor the Architecture Tradeoff Analysis Method (ATAM) (Clements et al. 2002) that is more flexible in the possible evaluation techniques, and various methods focusing on specific quality attributes such as modifiability (Bengtsson 2002).

Part of the evaluation in ATAM may be done by using Attribute Based Architectural Styles (ABAS), which is basically a description of a particular architecture style aimed at a particular quality attribute. Hence, an ABAS includes a certain architecture style and a specific method for evaluating architectures based on this style with respect to a certain quality attribute.

Another method that uses scenarios is Global Analysis (Hofmeister et al. 2000). In Global Analysis, scenarios are used to drive the design of the software architecture forward through different perspectives. Global Analysis is primarily aimed towards the creation of a software architecture than evaluation of software architectures, but this is done through continuously evaluating what has been done before.

As mentioned, all of these methods are focused on assessing the potential different software architecture candidates have for fulfilling a certain specification or blend of quality attributes. The method evaluated in

this paper is in this respect not different. There are, however, some aspects that differ between the aforementioned methods and the method evaluated in this paper.

While the aforementioned methods tend to be performed as a group activity, the method used in this paper allows each participant to first form his or her own opinion, and then the method facilitates in focusing discussions around those issues where the involved parties have different opinions. This means, of course, that issues that all participants agree on are not covered, thus focusing on the areas where opinions are different, and that the outcome is an increased joint understanding of the benefits and liabilities of the different architecture candidates, what needs to be further investigated, and how to go ahead with development.

Moreover, the methods above are focused on evaluating a single architecture to find out if and where there may be problems in it. The method evaluated in this paper is more aimed towards finding out which architecture candidate, of a set of architecture candidates, has the most potential to support the mix of quality attributes for a particular system to build. Accordingly, the method used in this paper does not produce any absolute measures on the architecture candidates. Rather, it produces judgements relative to the other architecture candidates on how good or bad the architecture candidates are with respect to a particular quality attribute, much in the same way as (Morisio et al. 2002) compares software artifacts with predefined ideal artifacts.

A final difference is that the method in this paper is not based on scenarios as the aforementioned methods are (except for (Morisio et al. 2002)). This may, however, be a possible and interesting future extension.

3. PROBLEM STATEMENT

In (Svahnberg et al. 2002) we simply assume that the vector sets in step two of the method (described in Section 1.2) can be created. In this paper we investigate whether it is at all possible to create these vector sets. This investigation consists of two parts:

1. It is generally held that some architectures are better at certain tasks than others. For example, architectures based on the Layered (Buschmann et al. 1996) architecture style are considered to be bad at performance but good at modifiability. Our first task is to find out if these generally held opinions in fact are so generally held, i.e. whether people will agree on the strengths and weaknesses of different architectures (or more specifically the architecture styles).
2. The second task is to find out whether there actually are any grounds to use a particular architecture in favour of another, or whether all architectures are in fact perceived equal and that the quality attributes a system presents are the results of an engineering effort. As previously mentioned, in this study we use generic architecture styles instead of the concrete architecture candidates for which the architecture evaluation method is intended. The reason for this is that with the abstract styles, there is no target system or target domain that imposes any special traits on the architecture candidates. Hence, if differences between the architecture styles are found, we can be more certain that these differences are in fact manifested in the styles and not quirks of the target domain.

The overall goal of the study is thus to investigate a specific approach for creating the individual frameworks used in the method outlined in Section 1.2. This approach should elicit the understanding of different software architecture candidates with respect to a set of quality attributes.

3.1 Research Questions

The research questions addressed in this paper are the following:

Q11. Is the perception of the strengths and weaknesses of different architecture candidates with respect to different quality attributes the same among a set of subjects?

Q12. Is the perception of which architecture candidates that best fulfil different quality attributes the same among a set of subjects?

These two questions correspond to the first task set out in Section 3, to seek an answer to whether it is at all possible to create an agreement among persons with different backgrounds. The answers to these two questions determine how accurate the answers to the following questions will be:

Q21. Is the perceived influence of different quality attributes ranked differently for different software architectures?

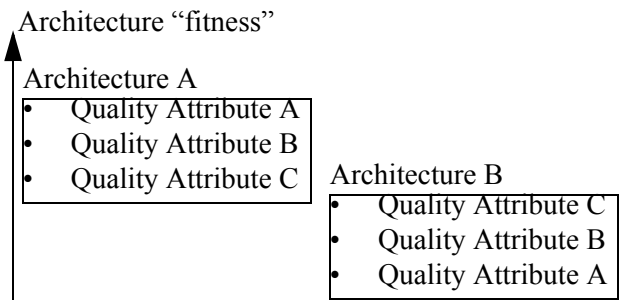

Figure 2. Ranking of two Architectures

Q22. Are software architecture perceived as supporting different quality attributes differently?

Q21 and Q22 correspond to the second task in Section 3, to find out whether stakeholders perceive large enough differences between different architecture candidates in terms of support for different quality attributes to motivate the choice of a particular architecture candidate over the others. The purpose of these two questions is thus to find out if we are able to quantify the perceived differences between architecture candidates.

The reason to divide each question (Q1x and Q2x) into two separate questions can be explained by examining the situation in Figure 2. In this figure, two architectures are rated compared to each other with respect to quality attribute C (the y-axis is assumed to represent a measure of architecture fitness or an absolute measure of support provided for different quality attributes). Moreover, within each architecture three quality attributes are rated. Just looking at the ranking of the quality attributes within each architecture one is lead to believe that architecture B is better than architecture A with respect to quality attribute C. However, when ranking the two architectures from the other perspective, i.e. with respect a certain quality attribute (attribute C in the figure) along the y-axis, it is clearly seen that even though quality attribute C is architecture B’s best quality it is still completely surpassed by architecture A. Hence, comparisons from both perspectives are necessary to get a complete picture.

4. STUDY DESIGN

In this section, we describe the planning and design of the study conducted to gather data to answer the questions posed in Section 3.1.

4.1 Context

The investigation is conducted in a university setting with academic people with a good knowledge of software architectures and software quality. In many empirical studies the university setting would be regarded as a validity threat. We prefer to see it as a step along the way, as discussed in Section 1.1. The contribution of this paper is in the evaluation phase, which is preferably performed in a lab environment. Based on what is learnt here, one can either move back to the innovation phase to further understand the phenomenon and evolve the theories, or move on to the application phase where the theories are tested in real life settings. Hence, the context of the study differs from that of the architecture evaluation method (as described in Section 1.3). In this study we focus on some particular aspects of the architecture evaluation method, and thus construct a setting where these can be measured in a controlled way. The results from this study are not intended to be usable in the full context of the architecture evaluation method.

4.2 Variables

Two sets of variables are used in the study. The first set is related to which software architecture candidates to include in the study, and the other concerns which quality attributes to use. From the perspective of the method any quality attributes and architecture candidates can be used. In an industrial setting, i.e. where the method is intended to be used, the architecture candidates would be concrete architecture suggestions for a system to design or re-evaluate and the quality attributes would be those relevant for this particular system and for the company as a whole.

In this study, however, we evaluate whether it is at all possible to discern between different software architectures with respect to quality attributes. In order to avoid that the quirks of a particular problem space clouds the results, we choose to use generic software architecture styles instead of concrete architecture candidates. These architecture styles are patterns, or templates, from which a concrete software architecture can be built. The architecture styles used in this study (presented below) are well tested styles with well known benefits and liabilities. The question to answer in this study is thus whether these benefits and liabilities are generally agreed upon and comparable.

We set no restrictions on the size of the software entity in which the software architecture styles are used. The

software entity may be an entire product, but may also be a module, a subsystem, or a component within a software system. The same goes for the selected quality attributes: in this study we have chosen the very top-level categories of quality attributes, but there are no restrictions in applying the evaluated method to lower-level, more concrete, quality attributes.

Architecture Candidates. In this study we use the following five architecture patterns of the eight patterns presented by (Buschmann et al. 1996), namely:

- Layered
- Pipes and Filters
- Blackboard
- Model-View-Controller
- Microkernel

We choose to exclude the patterns Reflection, because of its relative high complexity in comparison to the other patterns, Presentation-Abstraction-Controller, as it can be seen as merely a higher-level usage of the Model-View-Controller pattern, and Broker, as it is just a solution to a distributed problem that would if the problem was not distributed be represented as lower level design patterns (Gamma et al. 1995) and not as a system structure.

The benefits of choosing the architecture styles above is that they are familiar to most software engineers. They also represent the same level of abstraction, i.e. system or subsystem level. Being on this level of abstraction also ensures that it is possible to reason about quality attributes for each of the architecture styles. This is already today done in an informal way e.g. when deciding which architecture style to base a software architecture for a particular system on.

These architecture patterns can for the sake of this study be replaced with other styles, e.g. those presented in (Shaw & Garlan 1996). Likewise, the Design Patterns from (Gamma et al. 1995) also possible alternatives to use.

As presented in Section 5.1, for each of the patterns we provide the participants with copies of the text from (Buschmann et al. 1996) presenting an abstract of the pattern, an example, the context, the problem, solution and structure of the patterns. We avoided including the section covering benefits and liabilities of each pattern, in order to elicit the participants' own interpretations and experiences of the architecture patterns.

Quality Attributes. For the same reason that we use architecture styles in this study instead of concrete architectures, we also use generic quality attributes, not directly aimed at a particular software system. In order to get an as disparate set of quality attributes as possible, we choose to use one of the many different categorisations of quality attributes (see e.g. (McCall 1994)), i.e. the ISO 9126 standard (ISO 9126).

As indicated, one purpose of a classification of quality attributes is to provide a wide coverage of different aspects of a software system. The classification of these aspects thus focus on ordering the aspects in to high level groups where each group contains a set of aspects that logically belong together. It is the goal of every classification to try to maintain disparate sets of characteristica, or at least reduce the amount of overlap between different categories. Because of interactions between different quality attributes, this has proven to be a considerable challenge, and it is likely that categories of quality attributes in any categorisation will interact and support or compete with each other. However, we argue that it is still possible to compare different architecture candidates based on quality attributes - be they high level categories of quality attributes or low-level specific quality attributes. Q21 and Q22 aims at providing support for this claim.

For this study, with its academic context and the purpose of evaluating the potential of a particular architecture evaluation method, we choose to use the well known and generic quality attributes defined by the ISO 9126 standard (ISO 9126), categorised into the following main categories:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

The set of quality attributes used in this study can easily be replaced with another set, should there be any objections to the ones defined in this ISO standard. The intention is to select a well known set of quality attributes and then it is more important to study the research questions based on these, than to ponder whether these are an optimal selection.

We provide the participants with copies of pages 7 thru 11 of ISO 9126 (ISO 9126), where the high-level cate-

Table 1. Experiences of Subjects

Name	Title	Experience
Bob	Ph.D. 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	Ph.D.	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	Ph.D.	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	Ph.D.	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	Ph.D. Student	Ph.D. 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	Ph.D.	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.

gories of quality attributes and the lower-level quality attributes these contain are presented.

There are many quality attributes (and parts of some of the chosen quality attributes) that are not represented in the software architecture. Such quality attributes may have impact on development lead time, organisation structure, cost and so on. The architecture evaluation method studied in this paper focus on those aspects that are discernible by examining the software architecture, leaving other aspects to other methods. Hence, this method provides only one aspect necessary to make an informed decision, and it should be used together with other methods to create a full picture before a decision is taken.

4.3 Subjects

We choose to use a small board of experts in the field for the study, i.e. people from academia. More particularly, we use our colleagues and ourselves, which amounts to eight participants. Most of these participants have considerable industrial experience as well and some are even part time employed by industry, which ensures that the knowledge of the participants is not only based on theory, but also grounded in practise. Table 1 lists a brief description of the experiences of each participant[1]. The practical experiences of the participants range from large scale parallel applications to large scale database systems, general information systems, telecommunications applications, compilers and embedded applications.

None of these participants have anything invested in the results of this study, not even ourselves. There is a risk that the participants are biased in favour of previously used architecture styles, but considering the broad range of previous experiences from different application domains, a bias from one participant for a particular architecture style is levelled out by the other participants. Moreover, this would only affect one half of the results (covering the comparison of architecture structures with respect to each quality attribute, as discussed in Section 3.1 and in Section 4.5). The answers from a small number of biased participants would also be visible during the analysis of the results, as outliers.

1. Please note that the names are not the real names of the participants.

Hence, we can expect an objective result but are prepared to manage biased results.

4.4 Decision Method

When creating the individual frameworks, as outlined in Section 1.2, we need a decision method that enables a quantitative comparison between different quality attributes and architecture candidates. One such decision method is the Analytic Hierarchy Process (AHP) (Saaty 1980; Saaty & Vargas 2001), used in multi-criteria decision making and management science (Anderson et al. 2000). AHP is further presented in Appendix A.

AHP does not measure any absolute qualities of the software architecture candidates. Instead, it structures the experiences and knowledge of individuals with respect to the effect different architecture solutions have on different quality attributes. For the intended usages of the method in this study, we argue that this relative assessment is ample. According to (Helmer & Rescher 1959), using AHP need not even be a breech of objectivity. This may, however, be of lesser importance when considering the use to which the created data sets are put in the studied architecture evaluation method. As the data sets are used to drive discussions with the purpose of eliciting any problematic or misinterpreted issues, subjectivity in the form of personal experiences and background is in fact an almost desired quality, when applying the architecture evaluation method in a real life context.

4.5 Instrumentation

Each participant gets a different form, where the order of the questions is randomized according to three simple principles:

- One half of the subjects starts with questions related to Q11 and Q21. The other half starts with questions related to Q12 and Q22.
- For each of the two main questions (Q11+Q21 and Q12+Q22, respectively), the order of the architecture candidates and quality attributes is randomized.
- Finally, the questions (i.e. the pair-wise AHP comparisons) for each architecture candidate and quality attribute are asked in a random order.

Using these simple principles the questions are answered in a random order. The main reason of course being that the results should be independent of the order on the forms.

4.6 Validity Evaluation

In this section, the threats to the investigation are discussed. For a thorough description of possible threats to studies, see (Wohlin et al. 2000).

Conclusion Validity. As the answer to each question in the form is a subjective judgement, the answers will not be exactly the same for all participants. Indeed, it is not even certain that a participant will answer exactly the same, should the study be repeated. The reliability of each person's individual measures is thus not as high as one would hope. However, the consistency index within the AHP method helps to check the consistency of the individuals. Hence, as it is part of the study to measure the amount by which different participants disagree, as long as each participant is internally consistent this is not a problem.

The questions to answer are of such a character that the answer to each question is independent of when the previous questions were answered. Because of this we see no troubles if the exercise is interrupted for shorter time intervals, e.g. by e.g. phone calls, coffee breaks or fire drills. If, however, the breaks are longer, e.g. overnight, this may influence the consistency of the answers. This because it is possible to mature while completing the study, and one may sometimes wish to go back and correct some earlier answers, provided these are still fresh in memory.

If, during coffee breaks the participants discuss their answers, this may lead to participants changing their opinion on some questions. However, as a subsequent step in the studied method is to hold a discussion meeting to create a collective result (as outlined in Section 1.2), the only result of this would be to facilitate the process of creating a joint understanding.

As discussed in Section 4.3, the participants may be biased based on their previous experiences to favour a particular architecture structure. Because of the wide range of different experiences, we expect this effect to be equal for all architecture candidates and hence not visible. Moreover, if a participant favours a certain architecture structure when the general consensus have a different opinion this will be visible as outliers during the analysis.

The lack of a concrete and common target system may cause the participants to envision a target system based on their respective experiences when evaluating the software architecture candidates. There is a risk that

this may lead to a bias, as discussed above, in favour of the most common architecture style for each envisioned target system. This may also mean that the architectures are assessed in light of the characteristica of that particular target domain. However, because it is a separate part of the architecture evaluation method to also rate the quality attribute requirements for the target system, the assessment of the architecture candidates should be done without considering the typical requirements from the target domain.

Hence, envisioning different target systems and thus different instantiations of the architecture styles should not influence the results unduly. This is of course different between this study and the intended context of the method, where there indeed are concrete architectures and it is with a particular target system in mind that the architecture candidates are assessed. Moreover, because of the wide range of experiences as discussed above, we expect the envisioned target systems, if any, to be as different as the architecture candidates favoured out of bias, and any bias stemming from an envisioned target system as easy to detect as the bias stemming from previous experience.

Internal Validity. As the participants answer more and more questions, we expect them to become more acquainted with the AHP method, and possibly grow bolder in their answers, spreading them more from the centre value. To counter this effect, the forms are constructed in a random way, as described in Section 4.5. This ensures that questions asked last for one participant, i.e. when the participant has matured, are potentially asked first for another participant, when this person is still unmatured.

Construct Validity. We see no threat if the participants are aware of the hypothesis we try to prove, and hence make no secret of it. As we have no interest in favouring any particular architecture candidate or quality attribute, but rather to get a view of the participants opinions of all the architecture candidates and quality attributes, there is no way in which the participants can tweak their answers to satisfy any hidden hypothesis we may have.

The risk of bias based on experience to envision a target system or favour a particular architecture candidate, as discussed above, is in the context of this study reduced since the participants are themselves professional researchers trained at assessing alternatives objectively. Moreover, not having to actually build a system based on the assessment also increases the probability that the participants can relax and answer in an unbiased way. If anything, we expect the participants to be more critical towards the architecture candidates and the quality attributes they are familiar with. It is, of course, a different situation if the method is applied in an industry context with industry participants and the “threat” of having to build a system based on the architecture assessment.

We are aware that the method used in this study can also be used to evaluate persons, e.g. to see how close to some company norm their answers fall, and that some persons may perform differently as the result of knowing that they are being evaluated. However, we do not expect this to be a problem for the study in this paper. The participants are aware that this is not a personal evaluation, and are all professional researchers, aware of the intended usage of their answers.

External Validity. The construction of the study is such that it should work just as well at another place with a different set of participants with different backgrounds, a different set of quality attributes and a different set of architecture candidates.

5. OPERATION

In this section we describe the execution of the study, starting with the preparation, and continuing with the actual execution.

5.1 Preparation

Before the meeting, information was handed out to the participants concerning quality attributes and architecture candidates. This was handed out to ensure that people had approximately the same view on the architecture candidates and quality attributes, although we knew that most of the researchers were familiar with both the architecture styles and quality attributes anyway.

Specifically, pages 7 thru 11 of ISO 9126 (ISO 9126) was handed out as information regarding quality attributes, and a selection of the pages describing the patterns used in (Buschmann et al. 1996). The full text for the patterns is 100 pages, which would introduce a threat that the material has not been read before the meeting. Instead, the first pages of each pattern was used, presenting an abstract of the pattern, an example, the context, the problem, solution and structure of the patterns. We considered including the consequences-

section for each pattern, where the benefits and liabilities of using a particular pattern is presented, but opted not to, as this may introduce a threat that it is not the participants judgements of the patterns that emerge during the study but that of the authors of the patterns book.

Moreover, material presenting the decision method used, i.e. AHP, and a short introduction to the study was handed out. The brief description of AHP is to be seen more as background reading.

If the full method is applied, the preparations would also consist of creating the architecture candidates and describing the quality attributes used, before sending this out to the participants to study.

5.2 Execution

Based on previous experiences with AHP (Karlsson et al. 1998) we estimated that it would take approximately 20 seconds per question. Assuming six quality attributes there will be 15 (n*(n-1)/2) questions asked for each architecture candidate. With five architecture candidates we have 75 questions. Moreover, as we wish to consider both the comparison of quality attributes per architecture candidate (Qx1) as well as the comparison of architecture candidates with respect to each quality attribute (Qx2) so the number of questions is increased by another 60 questions ((5*(5-1)/2)*6), totalling 135 questions. This results in approximately 1½ hours of concentrated work for the participants.

As it turned out, the participants preferred to fill in the forms in their own offices, so after the introduction most of the participants left the meeting room and came back with the filled in form later. All participants had handed in the form after 1 ½ hour, which indicates that they worked relatively undisturbed. Hence, we do not believe that this introduced an extra threat to the study.

Moreover, during the introduction the following questions were defined as being the ones to answer:

- Which Quality Attribute, A or B, is most supported by the Architecture Candidate, and how much more supported is it?
- Which Architecture Candidate, A or B, is most fulfilling the Quality Attribute, and how much more does it fulfill it?

The texts on AHP presents the different answers as different grades of importance, e.g. “more important”, “highly more important” or “extremely more important”. In our study, we defined the answers to be “more supported”, “highly more supported” and “extremely more supported”.

5.3 Data Validation

As one of the research questions is to find out whether people agree or not, it seems unreasonable to disqualify some participants because their answers are not in conformance to the others, although people with disparate opinions were observed extra carefully during the analysis as this may for example indicate bias in favour of a particular architecture style.

AHP provides a consistency ratio that can be used to determine whether the participants have answered consistently, i.e. in agreement with themselves. Studying the consistency ratios, the participants were surprisingly consistent. Not one exceeded a ratio of 0.13, only one exceeded 0.12, most participants achieved a ratio of around 0.11 and one even scored a ratio of 0.03. These are very good values and indicate that the participants have indeed been consistent while completing the forms.

6. ANALYSIS AND INTERPRETATION

The data from the forms are, in accordance with the AHP method, reformatted into 11 vectors (one for each architecture candidate and one for each quality attribute) per participant. These 11 vectors per participant are used as input to the analysis phase.

6.1 Analysis for Q11 and Q12

In the analysis for Q11 and Q12, we identify the amount of agreement or disagreement for each of the 11 vector sets. This is important to answer the question whether it can be expected that people with different backgrounds can come to a general agreement. If this turns out to be the case, this also means that it is possible to easily identify participants with an opinion that differs from the majority. It is important to discuss such a difference of opinion and understand the cause of it before development commences. For example, differences in opinion may indicate that further investigations are necessary or that an important aspect has been overlooked. Hence, identifying where participants have different opinions is part of the preparations for the subsequent discussion meeting as described in Section 1.2.

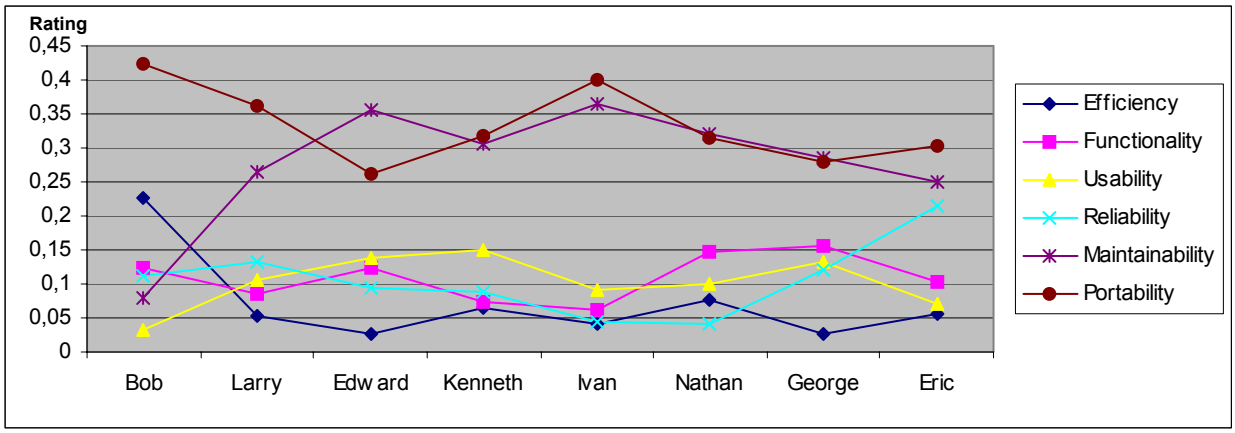

Figure 3. Layered architecture

Table 2. Number of persons per group and vector set

Vector	singles	2-groups	3-groups	4-groups	5-groups
Blackboard	1	2	1		
Layered		2		1	
Microkernel	3	1	1		
Model-View-Controller	1	1			1
Pipes and Filters	1		1	1	
Efficiency		2		1	
Functionality	3	1	1		
Maintainability	4	2			
Portability	1		1	1	
Reliability	2		2		
Usability	1	1			1

Comparing each vector set for all participants, the following four scenarios are possible:

1. All participants agree.
2. Most participants agree.
3. The participants form separate groups.
4. The participants disagree and form no groups.

Rather than setting some arbitrary thresholds as to what would constitute a group or not, we chose for this study to use principal component analysis (PCA) (Kachigan 1986) to find groups among how the participants have answered.

For the 11 vectors, this technique arranges the participants into groups in all cases, resulting in the values in Table 2. This table displays the number of groups of agreeing people and the sizes of these groups for each architecture candidate and quality attribute. For example, for Model-View-Controller and Usability five of the participants were in agreement, two more agreed with each other, and a single person did not agree with anyone else. Based on these values, it would be possible to conclude that cases 1 and 4 never occurred: never did all participants agree or disagree.

However, PCA does not take into account how small the differences between groups are. If all participants mostly agree, the technique simply becomes that more sensitive to smaller discrepancies. For example, the Layered architecture was expected to result in one, or at most two groups. The grounds for this assumption can be found by visually examining the values of the participants plotted into a graph. This graph is presented in Figure 3. As can be seen, there is a large consensus that Maintainability and Portability should score high, whereas the remaining quality attributes only have a minor role to play in the Layered architecture. However, PCA identifies several groups, consisting of:

- Edward and George
- Bob and Eric
- Kenneth, Ivan, Larry and Nathan.

These groups are not obviously identifiable in the graph in Figure 3. Hence, we conclude that we need a different way to measure where the participants agree or not.

To come to terms with that PCA sometimes finds too many groups, we need a number on how far away from each other the participants really are. For this we use the sum of the square of the distance to the mean value, according to the following formula:

$$\sum_{attributes} \sum_{persons} (x_i - \bar{x})^2$$

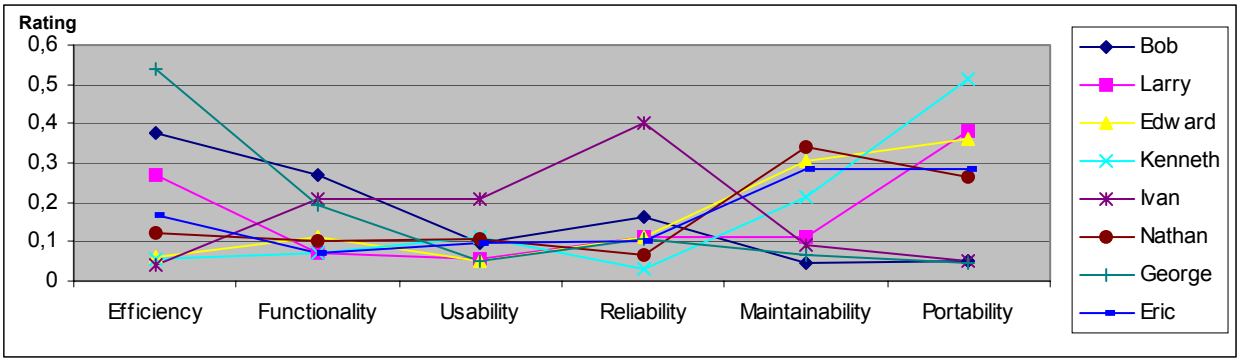

Figure 4. Microkernel architecture

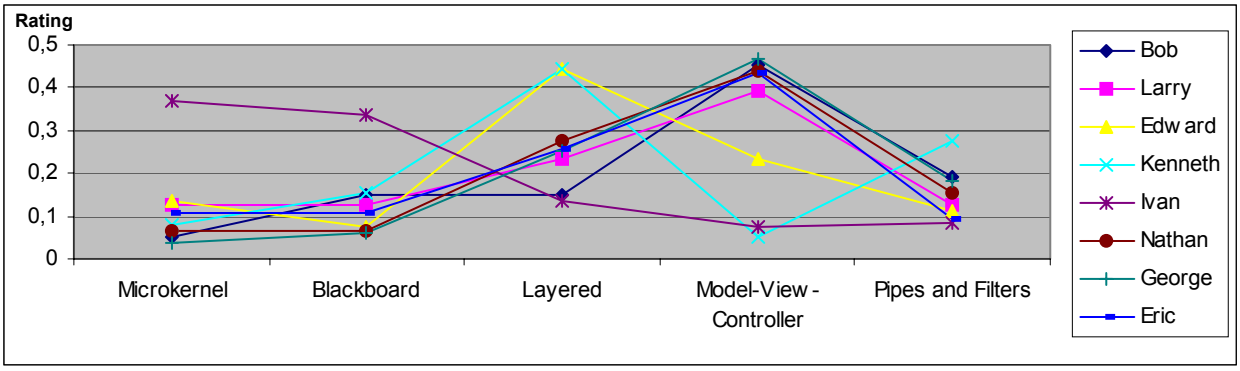

Figure 5. Usability

Table 3. Sum of squared distance to the mean

Vector	sum of squared distance to mean
Blackboard	0.505
Layered	0.150
Microkernel	0.693
Model-View-Controller	0.373
Pipes and Filters	0.385
Efficiency	0.453
Functionality	0.495
Maintainability	0.519
Portability	0.435
Reliability	0.592
Usability	0.468

Where *attributes* are the 5 architecture candidates or the 6 quality attributes that are compared within one vector set. These sums are presented in Table 3. This table shows a measure for each of the 11 vector sets a total over all data points in the vector how much the opinions differ.

We had hoped that these two measures would together give an indication of how well the participants agreed. However, much to our dismay many of the vector sets which we by examining the graphs would judge as being in agreement still score high values. By examining the graphs further, identifying the groups indicated by the PCA, it becomes clear that many high values can be explained by a few participants or groups of participants with extremely different opinions compared to the others. This is the case with for example the Microkernel architecture (Figure 4) and the quality attribute Usability (Figure 5).

The values for Microkernel are presented in the graph in Figure 4. In this case, we see that the largest discrepancy, which is also the likely basis for the PCA to divide the participants into two relatively large groups is that there are two disparate opinions regarding Maintainability and Portability.

For Usability, there is almost complete agreement, which is also indicated by the PCA and can be visually confirmed in the graph in Figure 5. Two persons, Edward and Kenneth have a slightly different opinion, albeit in agreement with each other, and a third participant, Ivan, is of a completely different opinion compared to the others.

Studying graphs of the vectors, such as the ones presented in Figure 3, 4 and 5, it is clear that PCA is too sensitive for our needs, and hence we instead recommend that the squared distance to the mean value is used for each data point (i.e. not summed over all attributes) to identify those points where there are disagreements so that discussions can be held around these points.

To continue the analysis of the answers, when taking outliers such as the ones visually identified in the graphs and with the help of the calculations above into account, there is mostly agreement between the participants on six of the eleven vector sets (i.e. Layered, Model-View-Controller, Pipes and Filters, Efficiency, Portability and Usability), and on three more (i.e. Blackboard, Microkernel and Functionality) the participants are reasonably in agreement, albeit forming separate groups. In two cases (Maintainability and Reliability) there is enough spread between the answers to claim that there is mostly disagreement among the participants.[1]

This means that there is a reasonably good joint understanding of several of the software architecture styles and quality attributes used. One major result here is that we are able to pinpoint (in a quantifiable way) where people agree and disagree, which can be used to spark discussions to resolve the disagreements during the subsequent discussion meeting described in Section 1.2.

Although not as important as the general agreement or disagreement, we also wanted to study whether certain individuals often had the same opinion. Thus, the next step in this analysis is to see how often different people agree. To this end we count the number of times two persons appear in the same group (i.e. agree on a vector set) for all 28 possible combinations of the eight participants. This count is found in Table 4.

We consider any group appearing five times or more to be in relative agreement. There are seven such groups, whereas a rough calculation yields that, statistically, a completely random ordering of persons into the same types of groups that PCA renders would give eight groups appearing five times or more. Hence, this may be a statistical artifact. On the other hand, it may not be that important whether certain individuals have the same opinion; it is most likely more important to understand where there seem to be a general agreement and where there is not.

1. For reasons of brevity we choose not to present all data sets in this paper. However, the data can be obtained by sending an e-mail to the first author.

Table 4. Number of groups that occur a number of times

Number of times	Number of groups
One time	1
Two times	9
Three times	6
Four times	5
Five times	2
Six times	4
Seven times	1

6.1.1 Summary

Q11 and Q12 concerns whether it is possible to create an agreement between subjects with different backgrounds. Using PCA we study how different people are grouped together and how many persons there are in each group. To come to terms with that PCA becomes more sensitive when there are smaller differences, we also study how far from the mean value the participants are. This is used as input when visually studying graphs of the vector sets.

After accounting for outliers, we conclude that there is an agreement for most of the architecture candidates and quality attributes, and more importantly that it is possible to identify participants with differing opinions. Although outside the scope of this paper, it is important to understand why participants have different opinions as there may be reasons to conduct further investigations or discuss the architecture candidates further to reach an increased joint understanding.

It should be noted that the sample size (eight participants) is not as large as one would hope for. This means that a small number outlying data points constitute a rather large part of the data set. Hence, we encourage replications of this study, if possible with larger sample sizes.

To answer questions Q11 and Q12, the perception of architecture candidates with respect to quality attributes is mostly similar among our set of subjects.

6.2 Analysis of Q21 and Q22

As the outcome of the analysis of Q11 and Q12 is mostly positive, it is meaningful to continue with an

Table 5. Correlation between software architectures

Comparison	Correlation
Microkernel vs. Blackboard	-0.0744
Microkernel vs. Layered	0.650
Microkernel vs. Model-View-Controller	-0.423
Microkernel vs. Pipes and Filters	0.565
Blackboard vs. Layered	-0.0168
Blackboard vs. Model-View-Controller	0.368
Blackboard vs. Pipes and Filters	0.528
Layered vs. Model-View-Controller	0.364
Layered vs. Pipes and Filters	0.351
Model-View-Controller vs. Pipes and Filters	0.145

analysis of Q21 and Q22, which is done below. We would, however like to point out that analysing different architecture patterns and abstract quality attributes is not the main intended usage of the method in this paper. Rather, in an industrial setting, the elements to analyse would be concrete architecture candidates for a particular software system, and the relevant quality attributes for this system. The analysis below should hence be seen more as an example of how to analyse the collected data, and the discussions held are examples of conclusions one can draw from the analysed data.

For the analysis of Q21 and Q22, the mean values of the participants results are used to create a unified view of the 11 vectors.

6.2.1 Analysis of Q21

For this analysis we calculate the correlation between the different architecture candidates, as it is simpler to calculate than PCA while still providing sufficient information.

We use a standard Pearson correlation, and the numbers that would indicate that there is a similarity between the ranking of quality attributes for an architecture candidate are large numbers, positive or negative. Large positive numbers indicate a perceived similarity between architecture candidates, whereas negative numbers would indicate that two architecture candidates are perceived to have tendencies to be mutually exclusive with respect to the quality attributes they support. A number near zero would indicate no perceived similarities between the architecture candidates.

The correlations between the vector sets for the software architecture styles are presented in Table 5. Based on this data (visually confirmed in graphs), we see that the Microkernel, Layered and Pipes and Filters architecture styles are perceived to have similar pros and cons with respect to the quality attributes they support. Slightly similar is also the Blackboard architecture.

One possible conclusion from this would be that one or more of these software architectures are perceived as unnecessary: other architectures are perceived to support almost the same mixture of quality attributes, and can hence be used instead. However, a more likely conclusion is that there are additional aspects and quality attributes, not included in this study, that differentiate between e.g. the Microkernel and Layered architecture styles. It would be an interesting future study to investigate exactly on what aspects these two architecture styles differ.

Moreover, there are very few negative relationships in Table 5. This may indicate that new software architecture patterns are needed that support an inverted mix of quality attributes as compared to the software architecture styles used in this study. This is not surprising, considering that the architecture styles (or patterns) presented in (Buschmann et al. 1996) are to some extent focused on a few quality attributes. For example, maintainability is in general rated highly for all the studied architecture styles from (Buschmann et al. 1996).

6.2.2 Analysis of Q22

This analysis is done in a similar way as for Q21 but with the six vectors of quality attributes, ranking the architecture candidates for each of the quality attributes. The correlations between the rankings are presented in Table 6. As can be seen, Efficiency and Functionality separates themselves nicely from all other quality attributes, indicating that the rankings of the architecture candidates for these are truly different than for the other quality attributes. What this means is that the participants in this study think that it will be difficult to combine requirements on Usability, Reliability, Maintainability and/or Portability together with requirements on Efficiency or Functionality when choosing among the software architecture styles included in this study.

For the other quality attributes (except for Efficiency and Functionality), we see that the participants perceive a high correlation between most of them, and

Table 6. Correlation between quality attributes

Comparison	Correlation
Efficiency vs. Functionality	-0.124
Efficiency vs. Usability	-0.812
Efficiency vs. Reliability	-0.421
Efficiency vs. Maintainability	-0.462
Efficiency vs. Portability	-0.251
Functionality vs. Usability	-0.404
Functionality vs. Reliability	-0.566
Functionality vs. Maintainability	-0.653
Functionality vs. Portability	-0.575
Usability vs. Reliability	0.721
Usability vs. Maintainability	0.588
Usability vs. Portability	0.404
Reliability vs. Maintainability	0.495
Reliability vs. Portability	0.805
Maintainability vs. Portability	0.719

this is visually confirmed when plotting their respective vectors together in a graph. This may indicate that these quality attributes are in fact related to each other, or simply recessive when compared to Efficiency and Functionality. The mostly high correlations between Usability, Reliability, Maintainability and Portability indicate that by selecting one of the architecture styles in this study that performs well on one of these quality attributes, it is believed that the other quality attributes may also perform reasonably well.

The opinions of the participants in this study regarding the quality attributes reconfirm the view expressed e.g. by (McCall 1994), that there exist quality attributes that are related to each other (either supporting each other or in conflict with each other). In (McCall 1994) this is derived in two ways: one is experience-based, and the other is by breaking down higher-level quality attributes to lower levels where it is found that many quality attributes depend on the same factors or metrics. This can be seen as a top-down approach. In our case, the conclusion is reached through a bottom-up approach, using the indirect evidence that the assessed software architectures are ranked similarly for the different quality attributes, and hence the quality attributes are at least not in conflict with each other, except for efficiency and functionality.

6.2.3 Summary

Q21 and Q22 concerns whether there is any perceived difference between different architecture candidates with respect to a set of quality attributes. To this end we study the correlation between the ratings given for the different architecture candidates and quality attributes.

For Q21, we conclude that the participants indeed perceive differences between the architecture candidates even if some are perceived as similar with respect to the support provided for the evaluated quality attributes.

For Q22, we conclude that for two of the quality attributes, i.e. efficiency and functionality, the sets of supporting architecture candidates are perceived to be very dissimilar compared to the other quality attributes. For the other quality attributes, the ranking of the architecture candidates seems to be fairly similar, meaning that the participants perceive no conflict in trying to support several of these quality attributes simultaneously.

In summary, we conclude that the architecture candidates are indeed different enough to warrant the choice of one candidate over another, given a particular set of quality attributes. However, except in some cases, the differences are not very large. This signifies that there are more factors that influence the success or failure of a software system than the choice of which architecture to use. The studied architecture evaluation method should thus only be seen as a small part of the overall design process, as one input among others for the decision makers.

As previously discussed, when deciding between different architecture candidates, it is also important to consider the required blend of quality attributes, i.e. the priorities of the quality attributes, since this ultimately decides which of the architecture candidates that best match the needs for a particular software system.

7. SUMMARY AND CONCLUSIONS

In this paper we evaluate a method for measuring the perceived amount of support different architecture candidates give for different quality attributes, and a way to rank different architecture candidates for any set of quality attributes. The main objective of this study is to show that it is possible to measure these things in the proposed way. To this end we have set up a number of research questions and conducted an empirical study.

7.1 Study Results

Based on the data analysis in Section 6 we draw the following conclusions regarding the research questions:

The first two questions, Q11 and Q12, seek an answer to whether it is possible to create an agreement among subjects with different backgrounds with respect to the strengths and weaknesses of a set of software architectures and the ranking of software architectures for different quality attributes. It is our belief that we make a strong argument that it is possible to obtain a shared perception about the strengths and weaknesses of different software architecture styles and their relative ranking for different quality attributes. We believe that it is easier to come to an agreement when using concrete software architectures rather than abstract architecture styles, and hence this result should be generalisable to concrete software architectures as well. However, there are discrepancies, and there is not as strong an agreement that could be expected, considering the ubiquity of the data set used.

The next question, Q21, concerns whether there is a perceived difference in the influence of different quality attributes for a set of architecture candidates. With a few exceptions (Microkernel, Layered and Pipes and Filters), the answer to this question is: yes, the different software architecture styles in our study are perceived to have disparate qualities, i.e. they are perceived to fulfil different quality attributes differently well. Moreover, there seem to be plenty of combinations of quality attributes that the participants think that no architecture candidate in our study supports very well, which means that there may be room for new software architecture styles that meet these combinations of quality attributes.

Lastly, for Q22, concerning whether software architectures support different quality attributes differently, the data clearly indicates that there is a considerable difference in the ranking of the architecture styles in our study, in particular between Efficiency and the other quality attributes and Functionality and the other quality attributes. For the other quality attributes, however, the differences in ranking of the architecture candidates are not so significant, being in some cases almost the same. The conclusion of this is that, except for when Efficiency or Functionality is required, some of the software architecture styles used in our study score high regardless of the quality attribute most wanted, whereas other software architecture styles are perceived to be bad at supporting all quality attributes in our study except for Efficiency or Functionality. The answer to Q22 is thus: yes regarding Efficiency and Functionality different software architecture styles in our study are perceived to support these differently well, and no regarding the other quality attributes there is no perceived major difference when the architecture styles are ranked in terms of provided support for the quality attributes.

7.2 Method Results

The answers to our research questions are such that we can conclude that it is indeed possible to use the method outlined in this paper together with AHP to measure the differences between software architecture candidates and quality attributes for a system to build or re-evaluate. More importantly, it is possible to identify areas where the participants disagree so that focused discussions can be held and an increased joint understanding can be reached.

Difficulties we have experienced while evaluating this method stem mainly from our choice of architecture candidates and quality attributes, in that we chose to use generic architecture patterns and generic ISO 9126 quality attributes. We believe that many of these difficulties would not arise if we applied the method in a company setting, letting developers in the company select between different architecture proposals for a software system. We are currently in the process of replicating this study in such a setting.

7.3 Data results

As a side result, we have during this study created two vector sets, the first comparing software architecture patterns with each other, and the second comparing quality attributes with each other. These vector sets are presented in Table 7 and Table 8. In Table 7, the architecture patterns are ranked for each quality attribute, and the table should hence be read row-wise. For example, Pipes and Filters, scoring 0.309 for Efficiency, is by the participants in this study considered to be able to support Efficiency almost three times as well as Layered, which only scores 0.110 for Efficiency. Naturally, these figures only express the participants' perceived potential of the architecture patterns. When instantiating concrete software architectures based on these patterns, the comparison may yield different figures. Moreover, when creating a software

Table 7. Ranking of Software Architectures per Quality Attribute

	Microkernel	Blackboard	Layered	Model-View-Controller	Pipes and Filters	Sum
Efficiency	0.274	0.186	0.110	0.121	0.309	1
Functionality	0.228	0.261	0.179	0.188	0.144	1
Usability	0.121	0.134	0.274	0.319	0.152	1
Reliability	0.207	0.103	0.270	0.239	0.180	1
Maintainability	0.124	0.171	0.283	0.198	0.225	1
Portability	0.194	0.0767	0.366	0.157	0.207	1

Table 8. Ranking of Quality Attributes per Software Architecture

	Microkernel	Blackboard	Layered	Model-View-Controller	Pipes and Filters
Efficiency	0.204	0.170	0.0714	0.0564	0.211
Functionality	0.138	0.284	0.109	0.199	0.152
Usability	0.0963	0.126	0.102	0.222	0.0980
Reliability	0.137	0.0711	0.106	0.132	0.132
Maintainability	0.182	0.254	0.278	0.230	0.251
Portability	0.244	0.0953	0.333	0.160	0.156
Sum	1	1	1	1	1

system it is necessary to consider a blend of quality attributes, which will assign different importances to the comparisons for the different quality attributes.

In Table 8, each architecture pattern contains a ranked list of quality attributes, and the table should hence be read column-wise. For example, the Microkernel architecture is considered best at portability, followed by efficiency and maintainability. It is perceived as almost equally good at functionality and reliability, and only slightly less good at usability.

These two vector sets form an initial framework for how different software architecture candidates may be compared and presented in terms of quality attributes. The tables, or rather tables created with a specific system in mind, provide valuable insights to make informed decisions regarding the selection of a software architecture for a system to build.

7.4 Implications of Data Results

As a result of this study, we see the following implications:

- There is not as much agreement as is generally held to be true regarding the support for quality attributes that different software architectures provide.
- There are several combinations of quality attributes that are perceived not to be met by any of the well-known software architecture patterns used in this study.
- With the exception of Efficiency and Functionality, our data indicates that quality attributes are, in general, not as mutually exclusive as is normally believed. This is also in accord with results presented in e.g. (Lundberg et al. 1999).

7.5 Future Work

First, we would like to encourage all readers to replicate this study, as the statistical power in our findings is limited. More subjects are needed and it is also essential to evaluate other software architectures and also more fine-grained divisions of quality attributes. However, the study provides some initial results and points to a potential way forward to informed decisions regarding architecture selection. We ourselves intend to conduct this study again, this time using industry people, to this end.

As the results for Q11 and Q12 show, there are points where people disagree. We intend to investigate whether the method can be used to pinpoint these disagreements as input to a discussion with the purpose of creating a joint understanding of the benefits and liabilities of the architecture candidates.

8. 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.

[Bengtsson 2002] PO Bengtsson, *"Architecture-Level Modifiability Analysis"*, Ph.D. Thesis, Blekinge Institute of Technology, Dissertation Series No 2002-2, 2002.

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

[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.

[Clements et al. 2002] P. Clements, R. Kazman, M. Klein, "Evaluating Software Architectures - Methods and Case Studies", Addison-Wesley, Boston MA, 2002.

[Gamma et al. 1995] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *"Design Patterns: Elements of Reusable Object-Oriented Software"*, Addison-Wesley Publishing Co., Reading MA, 1995.

[Helmer & Rescher 1959] O. Helmer, N. Rescher, "On the Epistemology of the Inexact Sciences", in *Management Science* **6**(1):25-52, 1959; reprinted in Leonard Krimerman (ed.), "The Nature and Scope of Social Science", Appleton-Century-Crofts, New York NY, pp. 181-203, 1969.

[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).

[Johansson et al. 2001] 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.

[Kachigan 1986] S.K. Kachigan, "*Statistical Analysis – An Interdisciplinary Introduction to Univariate & Multivariate Methods*", Radius Press, 1986.

[Karlsson et al. 1998] J. Karlsson, C. Wohlin, B. Regnell, "An Evaluation of Methods for Prioritising Software Requirements", Information and Software Technology, Vol. 39, pp. 939-947, 1998.

[Lindvall et al. 2003] M. Lindvall, R.T. Tvedt, P. Costa, "An Empirically-Based Process for Software Architecture Evaluation", in *Empirical Software Engineering*, **8**(1):83-108, 2003.

[Linkman & Rombach 1997] S. Linkman, H.D. Rombach, "Experimentation as a Vehicle for Software Technology Transfer - A Family of Software Reading Techniques", in *Information and Software Technology* **39**(11):777-780, 1997.

[Lundberg et al. 1999] L. Lundberg, J. Bosch, D. Häggander, PO Bengtsson, "Quality Attributes in Software Architecture Design", in *Proceedings of the IASTED 3rd International Conference on Software Engineering and Applications*, IASTED/ Acta Press, Anaheim CA, pp. 353-362, 1999.

[McCall 1994] J.A. McCall, "Quality Factors", in *Encyclopedia of Software Engineering*, J.L. Marciniak (Editor), John Wiley & Sons, New York NY, pp. 958-969, 1994.

[Morisio et al. 2002] M. Morisio, I. Stamelos, A. Tsoukiàs, "A New Method to Evaluate Software Artifacts Against Predefined Profiles", in *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering (SEKE 2002)*, ACM Press, New York NY, pp. 811-818, 2002.

[Parnas 1994] 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.

[Perry & Wolf 1992] D.E. Perry, A.L. Wolf, "Foundations for the Study of Software Architecture", in *ACM SIGSOFT Software Engineering Notes* **17**(4), 1992.

[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.

[Svahnberg et al. 2002] M. Svahnberg, C. Wohlin, L. Lundberg, M. Mattsson, "Quality Attribute-Driven Selection of 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, 2002.

[Wohlin et al. 1996] C. Wohlin, A. Gustavsson, M. Höst, C. Mattsson, "A Framework for Technology Introduction in Software Organizations", in *Proceedings of Software Process Improvement Conference*, pp. 167-176, Brighton, UK, 1996.

[Wohlin et al. 2000] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, *"Experimentation in Software Engineering"*, Kluwer Academic Publishers, Dordrecht, the Netherlands, 2000.

Appendix A - Analytic Hierarchy Process

Simply put, the Analytic Hierarchy Process (AHP) is a process for making pairwise comparisons. By making all possible pairwise comparisons it is possible to make a ranking of the entities and to, in addition, calculate a consistency index. AHP consists of four basic substeps and an added fifth substep to analyse the consistency in the results, described below.

Substep 1: Create an $n \times n$ matrix (denoted *N*), where *n* is the variables to be compared, for example, quality attributes. In the diagonal in the matrix the value "1" is inserted. The matrix is referred to as the comparison matrix. Element n_{ij}, when *i* is not equal to *j,* records the relative importance of variable *i* versus variable *j*.

Substep 2: Perform a pairwise comparison of the variables with respect to the importance. The scale for comparing the variables pairwise is illustrated in Figure 6. Each pairwise comparison means that it is necessary to determine which of two variables is most important and how much more important it is. For example, a marking to the left on the scale means that variable *i* is more important than variable *j*. The interpretation of the values is shown in Table 9.

The pairwise comparison is conducted for all pairs, i.e. $n \times (n-1)/2$ comparisons. The relative importance values are put into the matrix created in the first step and the reciprocal for each pair is determined and also put into the matrix. This results in a complete $n \times n$ matrix.

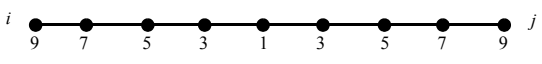

Figure 6. The scale for the AHP comparison.

Substep 3: Compute the eigenvector of the $n \times n$ matrix. A simple method is proposed by (Saaty 1980; Saaty & Vargas 2001) to do this; the method is known as averaging over normalised columns, and the procedure is as follows:

- Calculate the sum of the columns in the matrix, $n_j = \sum_{i=1}^{n} n_{ij}$.
- Each element in a column is divided by the sum of the column, $m_{ij} = n_{ij}/n_j$. This results in a new matrix, denoted *M*, with elements m_{ij}.

Table 9. Scale for pairwise comparison using AHP (Saaty 1980; Saaty & Vargas 2001).

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 10. Matrix order and corresponding RI value (Saaty 1980; Saaty & Vargas 2001).

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.45	1.51	1.48	1.56	1.57	1.59

- The sum of each row in the new matrix is calculated, $m_i = \sum_{j=1}^{n} m_{ij}$.
- The sums of the rows are normalised by dividing by the number of variables (n), $P_i = m_i/n$. This results in an estimation of the eigen values of the matrix, and it is referred to as the priority vector. The vector is denoted P with elements P_i for $i = 1 \ldots n$.

Substep 4: Assign a relative importance to the variables. The first variable is assigned the element in the priority vector. It is said that the first variable accounts for P_i percent of the importance of the variables. The second variable is assigned the second element in the priority vector and so on. Let P_1 to P_n be the percentage values for the importance of variables *1* to n.

Substep 5: Because AHP conducts more comparisons than minimally necessary, it is possible to evaluate the consistency of the ranking. This consistency ratio captures how consistently the pairwise comparison has been conducted. The consistency check is particularly important when a large number of pairwise comparisons are necessary, making it easier to make errors and hence become inconsistent.

The consistency ratio is computed in two steps.

- First, a consistency index (CI) is computed as $CI = (\lambda max - n)/(n-1)$, where λmax is the maximum principal eigen value of the $n \times n$ matrix. The closer λmax is to n the smaller is the error in the comparison. λmax is calculated by first multiplying the comparison matrix, i.e. matrix N, with the priority vector. Let the resulting vector be denoted R with elements R_i, $R = N \times P$ For the resulting vector, each element in the vector is divided by the corresponding element in the priority vector, $\lambda_i = R_i/P_i$. λmax is now computed as the average of the elements in the resulting vector, $\lambda max = \left(\sum_{i=1}^{n} \lambda_i \right) / n$. CI can now be calculated.
- The consistency ratio (CR) is determined by dividing the consistency index (CI) by a random index (RI). The random index has been generated to take into account randomness and it is used to normalise the consistency index. Thus, $CR = (CI)/(RI)$, where RI is determined from Table 10, where the first row shows the order of the matrix (n) and the second row shows the corresponding RI value. The smaller CR, the more consistent is the comparison.

According to (Saaty 1980; Saaty & Vargas 2001), a consistency ratio of 0.10 or less is considered acceptable. It is, however, pointed out in the literature that in practice higher values are often obtained. This indicates that 0.10 may be too hard, but it indicates an approximate size of the expected consistency ratio.