Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process

Liming Zhu limingz@cse.unsw.edu.au
School of Computer Science and Engineering, University of New South Wales
Empirical Software Engineering, National ICT Australia

Aybüke Aurum aybuke@unsw.edu.au
School of Information Systems, Technology and Management, University of New South Wales
Empirical Software Engineering, National ICT Australia

Ian Gorton ian.gorton@nicta.com.au
Empirical Software Engineering, National ICT Australia
School of Computer Science and Engineering, University of New South Wales

Ross Jeffery ross.jeffery@nicta.com.au
School of Computer Science and Engineering, University of New South Wales
Empirical Software Engineering, National ICT Australia

Abstract. Software architecture evaluation involves evaluating different architecture design alternatives against multiple quality-attributes. These attributes typically have intrinsic conflicts and must be considered simultaneously in order to reach a final design decision. AHP (Analytic Hierarchy Process), an important decision making technique, has been leveraged to resolve such conflicts. AHP can help provide an overall ranking of design alternatives. However it lacks the capability to explicitly identify the exact tradeoffs being made and the relative size of these tradeoffs. Moreover, the ranking produced can be sensitive such that the smallest change in intermediate priority weights can alter the final order of design alternatives. In this paper, we propose several in-depth analysis techniques applicable to AHP to identify critical tradeoffs and sensitive points in the decision process. We apply our method to an example of a real-world distributed architecture presented in the literature. The results are promising in that they make important decision consequences explicit in terms of key design tradeoffs and the architecture's capability to handle future quality attribute changes. These expose critical decisions which are otherwise too subtle to be detected in standard AHP results.

1. Introduction

A systems' software architecture embodies a collection of architecture decisions which respond to multiple quality attribute requirements (Bosch 2004). In order to assess to what extent the quality requirements have been met, software architecture evaluation needs to be conducted at various phases of the software development life cycle (Bass, Clements et al. 2003) (Bosch 2000). While a design decision could be adopted in favor of some quality attributes, it usually comes at the expense of others due to the competing nature of quality attributes (Klein and Kazman 1999). When conflicts arise, the priority of competing quality requirements must be elicited (Karlsson, Wohlin et al. 1998) and used in a process of requirement negotiation so a consensus can be reached on solving the conflicts (Svahnberg, Wohlin et al. 2002). However, for complex architectures, a Multi-Criteria Decision Making (MCDM) technique can provide more formal and quantitative reasoning. One such MCDM

technique, AHP (Saaty 1980), has recently been leveraged in selecting a desired architecture among candidates (Svahnberg, Wohlin et al. 2003).

Applying AHP in a standard manner can provide overall priority weights of design alternatives. It takes into consideration all quality attributes, priority weights of design alternatives for individual quality attributes and priority weights among the quality attributes themselves. However, if a design alternative is chosen using AHP, the key design tradeoffs made for each quality attributes and the relative sizes of the tradeoffs are not explicitly revealed. We consider these trade-offs important consequences of the final design decision; consequences that should be explicitly presented to stakeholders and re-examined if necessary.

AHP relies on pair-wise comparisons to acquire various priority weights among design alternatives and quality attributes. These lead to the overall ranking of design alternatives. In some cases, the smallest differences in these priority weights will alter the final ranking. These sensitive priority weights could be interpreted as follows:

1) Decisions leading to small weighting differences could be the most critical ones. Whether these decisions accurately reflect the real judgments of architects and stakeholders could require closer examination.

2) In a world where requirements are constantly changing, so are the priorities of quality attributers. It is important to know whether the new priorities will imply a different ranking. Small weighting differences that can lead to a ranking change indicate the sensitivity of the resulting ranking to change. Stakeholders could be informed of this and factor it into the design decision.

Standard AHP practice lacks the capability to explicitly identify the exact tradeoffs being made and the relative magnitude of these tradeoffs. Moreover, the ranking produced can be sensitive such that the smallest change in intermediate priority weights can alter the final order of design alternatives.

In this paper, we employ in-depth analysis techniques on results produced by a standard AHP to:

- Identify tradeoffs implied by an architecture alternative, along with the magnitude of these tradeoffs
- Identify the most critical decisions in the overall decision process
- Evaluate the sensitivity of the final decision and its capability for handling quality attribute priority changes

These results make decision consequences more explicit and illustrate the rationale behind decisions produced by the AHP method. Identifying critical decisions and performing sensitivity analysis can expose potential issues and lead to an architecture better prepared for future change. In addition, the techniques are not restricted to any particular domains.

We begin by discussing related work on tradeoff analysis and sensitivity analysis. In section 3 we describe the case study used and the standard AHP results in (Al-Naeem, Gorton et al. 2005). Our approach is explained in section 4. Section 5 presents a discussion and finally we conclude our paper in section 6.

2. Related Work

2.1 Software Architecture Evaluation and Decision Making

Software architecture evaluation has emerged as an important software quality assurance technique. The principle objective of evaluating architecture is to assess the potential of the chosen architecture to deliver a system capable of fulfilling required quality requirements. A number of methods, such as Architecture Tradeoff Analysis Method (Kazman, Barbacci et al. 1999) and Architecture-Level Maintainability Analysis (ALMA) (Bengtsson, Lassing et al. 2004), have been developed to evaluate the quality related issues at the architecture level.

Architecture evaluation can be seen as a phase of the decision-making process. A decision-making process consists of the following activities: problem identification; problem analysis and solution development; selection and evaluation. Though architecture evaluation focuses on selection and evaluation activities, it often covers solution development in an iterative process.

The outcomes of on-going architecture evaluation throughout a project lifetime can be positioned on a continuum between unstructured (open-ended) decisions and structured decisions (Aurum and Wohlin 2003). Architecture evaluation allows us to forecast the optimum quality attributes by dealing with uncertainties in both subsequent implementation technology and changing requirements. Hence, we consider architecture evaluation to be a decision-making process which contains open-ended components.

Most architecture evaluation methods conduct evaluation for individual quality attributes first and consolidate the results later. Attribute-specific evaluation requires reasoning models and expertise for the quality attribute in focus. Consolidation requires a decision making process for balancing tradeoffs and selecting the best candidates when quality requirements are conflicting. In this paper we focus on the latter, selecting the design alternatives, which is essentially the final stage of decision-making process. Open research questions remain on how these tradeoff analyses should be performed and how sensitive the final decisions are (Dobrica and Niemela 2002). In this paper, we propose several new analysis techniques to answer these questions by complementing existing methods in the context of AHP. To be precise, we identify critical tradeoffs and sensitive points in the AHP decision process.

2.2 Tradeoff Analysis

Various tradeoff analysis techniques in architecture evaluation have been proposed either separately or as a part of evaluation methods. Most notable are the tradeoff analysis in ATAM (Architecture Tradeoff Analysis Method) (Kazman, Barbacci et al. 1999), Cost Benefit Analysis Method (CBAM) (Kazman, Asundi et al. 2001), goal satisfaction analysis between different designs in the NFR (Non-Functional Requirements) Framework (Gross and Yu 2001; Liu and Yu 2001) and AHP related MCDM decision process (Svahnberg, Wohlin et al. 2002; Svahnberg, Wohlin et al. 2003).

2.2.1 Architecture Tradeoff Analysis Method

Architecture Tradeoff Analysis Method (ATAM) is a scenario-based architecture evaluation method. Quality scenarios are gathered through stakeholder workshops and requirement analysis. When inconsistency between different stakeholder viewpoints appears, negotiation or aggregation is used to obtain a final result. The gathered scenarios are used to assess an architecture's support for quality attributes. Architecture elements

affecting a quality attribute can be identified. If some of these elements affect different quality attributes simultaneously, these elements are defined as tradeoff points.

A tradeoff point could lead to a tradeoff decisions. When this occurs, ATAM largely leaves the decision process to requirement negotiation. Stakeholders can use the WinWin model to assist their negotiation in resolving conflicts following a rigorous process (In, Boehm et al. 2001; In, Kazman et al. 2001). These methods have proved useful when consensus can be reached among stakeholders(Svahnberg and Wohlin 2002).

When relationships among design alternatives and quality attributes are too intricate to solve by informal negotiation, more formal quantitative methods need to be used. Two types of method exist in the architecture evaluation domain. One is to use a method like Cost Benefit Analysis Method (CBAM) (Kazman, Asundi et al. 2001) in which stakeholders can link all potential design decisions to their benefits through a response-utility function and then perform a value analysis to determine the best candidates (Kazman, Asundi et al. 2001). In the real world, the utility function of the responses can be difficult to solicit from stakeholders (Bass, Clements et al. 2003). The other type of method is to use MCDM techniques, such as AHP. Weighted priorities are either derived from pair-wise qualitative comparisons or converted from more quantitative reasoning results. So far, all the practices of AHP in architecture evaluation are dealing with "quality response", not "business benefits" when pair-comparisons are conducted. Cost can be considered as one of the quality attributes as shown in our case study or considered later. However, if a final decision hinged on business benefits has to be made, the AHP's results regarding quality will need to be further reasoned by associating architecture quality with benefits.

Thus, in terms of trade-off analysis, ATAM is more suitable for initially identifying tradeoffs than for resolving them. If business benefits are the immediate concern of a particular architecture evaluation session and response-utility function can be solicited, CBAM should be used after ATAM. If the main concern of a particular architecture evaluation session is the overall quality (including cost if desired) of the architecture, current normal practice of AHP can be applied. Our approach provides crucial additional in-depth analysis on top of a standard AHP. We argue that methods like CBAM may also benefit from explicit quantitative tradeoff and sensitivity analysis because of the possible inaccuracy of the utility function and changing quality scenario priorities. Since our approach explicitly deals with AHP data, it is not applicable to CBAM.

On the other hand, it is possible to modify the current way of using AHP by associating weighted priority with its potential business benefits utility in the intermediate steps to enable business benefits interpretations of the final result. The implication of this modification comparing to CBAM is beyond the scope of this paper.

2.2.2 Non-Functional Requirement framework

Quality attribute requirements are often known as Non-Functional Requirements **(**NFR). Researchers at the University of Toronto applied goal modeling techniques to NFRs and design decisions (Chung, Nixon et al. 2000). In a goal model, a design alternative can be modeled as a task and a NFR can be modeled as a goal. Different levels of NFRs can be modeled as goals and subgoals. A design alternative (task) can positively or negatively contribute to NFRs (goals/subgoals). These contributions are modeled in the goal graph to reflect the tradeoffs made locally towards the immediate goals. In order to use NFR goal modeling as a successful decision-making process, evaluators need to have a solid understanding of the consequences of the solutions

regarding individual quality attribute. NFR provides a way of consolidating and balancing these understandings considering multiple quality attributes simultaneously.

In order to discover the final satisfaction level of a goal, contributions can be calculated and propagated through the goal graph. The qualitative contribution was later extended to include quantitative values in order to accommodate more accurate contributions (Giorgini, Mylopoulos et al. 2004). Although the theory has been proved and demonstrated through small case studies (Giorgini, Mylopoulos et al. 2004), it has not been applied systematically in real architecture evaluations due to the lack of dedicated tool support for propagation calculation. If the contributions of a chosen design alternative can be populated through propagation calculation, these values can be used as the starting point of an in-depth quantitative tradeoff and sensitivity analysis.

Thus, the method remains at the stage where users can produce a goal model to help them understand and clarify the tradeoffs qualitatively.

2.2.3 Analytic Hierarchy Process

Analytic Hierarchy Process (AHP), as a critical decision making tool for several disciplines, has proved controversial (Triantaphyllou and Mann 1994). The pair-wise comparisons, data normalization across different criteria and algorithms leading to priority weights can be misleading if care is not taken. The implication of this from a decision making perspective is that the evaluation stage of the decision becomes crucial after the selection process. Therefore, final decisions need to be re-examined to prevent potential mistakes from being propagated through algorithms whose implications are implicit.

In addition, AHP requires users to take a holistic view of the design alternatives while comparing them without taking into account the analysis and intermediate results leading up the alternatives. This tends to neglect the Solution Development stage in a decision making process so the implications of intermediate decisions and analysis are lost. Tradeoffs within a design alternative tend to be much less explicit. This holistic view may lead to situations where the final ranking hinges on sensitive and critical decisions of which users are not aware.

Several attempts (Svahnberg, Wohlin et al. 2002; Svahnberg, Wohlin et al. 2003; Al-Naeem, Gorton et al. 2005) have been made to incorporate AHP into architecture evaluation.

In (Al-Naeem, Gorton et al. 2005), researchers have proposed a systematic quality-driven approach for evaluating distributed software applications. Design alternatives were divided into different groups. Within each group, AHP was applied. It then used IP (Integer Programming) to consolidate the AHP results into a final ranking. No explicit tradeoff analysis was performed. However, the method has the potential to conduct future analysis which can reveal the intricacies of intermediate decisions within the whole decision-making process.

Applying AHP in architecture evaluation is best formalized in (Svahnberg, Wohlin et al. 2003). Not only does it produce a Framework for Quality Attribute (FQA) to capture architecture rankings according to their ability to meet particular quality attribute, but it also produces a Framework for Architecture Structures (FAS) to obtain quality attribute rankings for each architecture. FAS provides extra consistency checking and further adjustment in addition to normal AHP consistency ratio checking for pair-wise comparisons. The method also

provides confidence levels on the final ranking by providing a Framework for Variance Calculation (FVC). However, various priority frameworks do not provide critical tradeoff information directly.

2.3 Sensitivity Analysis

The question of sensitivity in architecture evaluation methods was first raised in (Dobrica and Niemela 2002), but no answer has subsequently been published. This is partially because most of the evaluation methods are qualitative. A sensitivity analysis for most methods might require extensive empirical studies involving conducting experiments.

AHP, on the other hand, is a quantitative method for performing architecture evaluation. It has the potential to answer the sensitivity question in tradeoff analysis stage. In (Svahnberg, Wohlin et al. 2003), for the final ranking, the method provides a variance calculation which indicates the certainty of the results. However, if the variance calculation exposes a potential uncertain ranking, there is no way we can easily identify the problem spots.

3. Case Study: Applying AHP

Our in-depth analysis is based on standard AHP data which is acquired by following the steps provided in AHP-based evaluation methods (Svahnberg, Wohlin et al. 2003; Al-Naeem, Gorton et al. 2005). Our approach simply provides additional analysis on existing AHP data sets rather than a new process. Thus we do not feel it is necessary to repeat those steps in our paper in a separate methodology section. We believe that it would be much clearer to elaborate the additional analysis techniques in the context of the existing data set. We choose to combine our technique section with the case study in this paper.

We base our in-depth analysis on the examples in (Al-Naeem, Gorton et al. 2005), which applies AHP to an architecture design for an application in (Gorton and Haack 2004). We look at how standard AHP results produced in the example can be further analyzed to examine tradeoff and sensitivity questions. We chose examples in (Al-Naeem, Gorton et al. 2005) instead of (Svahnberg, Wohlin et al. 2003) as the former considers several quality attributes and design alternatives, with the difference between certain alternatives subtle. This makes our analysis techniques more easily demonstrated. Although (Al-Naeem, Gorton et al. 2005) didn't use analysis methods as elaborated as (Svahnberg, Wohlin et al. 2003) with additional FAS and FVC analysis, the gist is the same. If we applied our method to (Svahnberg, Wohlin et al. 2003), the FAS adjusted priority weights would be used. Sensitivity analysis would identify precise critical decisions beyond the general certainty level provided by FVS.

AHP analysis can be performed using a spreadsheet tool like Excel. We complement our analysis with Expert Choice (2004) because it provides most of the required data automatically and includes useful visualizations.

The example application, the Glass Box (GB) project, used in (Gorton and Haack 2004; Al-Naeem, Gorton et al. 2005) is a deployed, complex distributed application which helps analysts to monitor, retrieve and analyze information in real time in a distributed and changing environment. Nineteen different architecture alternatives and architecturally significant design decisions with 171 valid combinations were considered and evaluated by extensively interviewing the project lead. These design alternatives were grouped into five groups. Within each

group, AHP was applied. The final results were produced using Integer Programming (IP) which is beyond the scope of this paper.

Among the five groups, we choose the Architecture (Arch) design decision group in this paper as it is the most crucial decision affecting the resulting core architecture solution. There are four design alternatives considered for this design decision:

1. Three Tier J2EE (THHJ)
2. 3-tier using .Net (THTD)
3. 2-tier (TWOT)
4. A distributed agent support platform (COABS)

Seven associated quality attributes are considered based on extensive interviews with the project lead and stakeholder requirements. They are:

1. modifiability
2. scalability
3. performance
4. cost
5. development effort
6. portability
7. ease of installation

The intermediate data and final results acquired through applying AHP are presented in Table 1 and Figure 1.

Each row in Table 1 is obtained through pair-wise comparisons of the four design alternatives in terms of their support for the quality attributes. The sum of the priority weights in each row equals 1 which is constrained by the AHP method. We reproduce the priority weights among quality attributes and overall design alternative ranking in Figure 1 using Expert Choice. On the left side, priority weights are obtained through pair-comparisons among quality attributes. The data is in percentage format. These weights are aggregated from different weights assigned according to goals from different stakeholders. Some goals are aligned with business goals. Some goals are aligned with development and costing goals. On the right side is the final overall ranking.

Table 1. Design Alternatives Ranking for Each Quality Attribute (Adapted from table 6 in (Al-Naeem, Gorton et al. 2005))

Quality Attributes	Design Alternatives			
	THTJ	**THTD**	**TWOT**	**COAB**
Modifiability	0.521	0.182	0.046	0.250
Scalability	0.402	0.402	0.054	0.143
Performance	0.204	0.204	0.347	0.246
Cost	0.166	0.120	0.487	0.227
Dev. Effort	0.152	0.110	0.515	0.223
Portability	0.450	0.050	0.050	0.450
Ease of Inst.	0.168	0.368	0.256	0.208

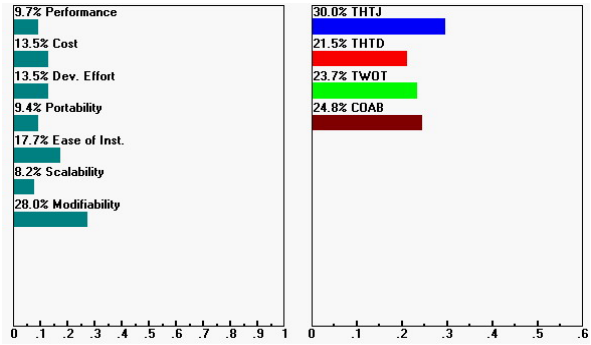

Figure 1. Weighted Priorities for Quality Attributes and Overall Design Alternative Ranking(Adapted from table 4 and 7 in (Al-Naeem, Gorton et al. 2005))

As we can see from Figure 1, modifiability is the most important quality attribute with a priority weight of 0.28, followed by ease of installation. In final design alternative ranking, THTJ outperformed all other candidates significantly. COAB and TWOT perform very closely with priority weights of 0.237 and 0.248 respectively. THTD was the last choice.

These standard AHP results tell us very little beyond a simple ranking of alternatives. In addition to a ranking, we would like to know the exact consequences of the chosen design alternative in terms of the key tradeoffs being made and the extent of these tradeoffs when compared to tradeoffs implied by other alternatives. For the final ranking, the close weightings of some alternatives does not make it clear whether slightly different intermediate decisions on previous pair-comparisons or priority weights could change the outcome.

4. Tradeoff Analysis and Sensitivity Analysis in AHP

4.1 Tradeoff Analysis

Tradeoffs are made between conflicting quality attributes. Since quality attributes are prioritized, we can look at tradeoffs in two different ways in terms of considering or not considering the priority weights. Without weights, the tradeoff reflects a more local point of view. A local point of view on tradeoffs is to represent the size of the tradeoff without considering the respective importance of the quality attribute. With weights, tradeoffs associated with the most important quality attributes are highlighted.

4.1.1 Tradeoffs without quality attribute weights

A traditional tradeoff in architecture evaluation is considered as the relinquishment of one quality requirement for another that is regarded as more desirable in a particular project context. The tradeoff size can be considered in absolute terms for each quality attribute. For example, in order to support the system on another platform (portability), we may choose a design/implementation alternative which supports a transaction throughput of 3000 TPS (transactions per second) instead of 4500 TPS (performance). The exact number is useful when a quantifiable non-functional requirement can be used for comparison.

However, most of the time, the reason to resort to AHP is that no easily quantifiable NFR exists. Therefore, tradeoffs must be considered in relative terms. For example, in order to support a higher modifiability priority (with 0.280 weights), we may choose a design alternative which entails a lower performance priority (with 0.097 weights).

In AHP, we treat tradeoffs relatively in terms of priority. Applying AHP, we can obtain the priority weights of design alternatives for each quality attribute as shown in Table 1. In addition, we can choose any two quality attributes and plot their priorities along the x and y axis. This can be achieved using a two dimensional sensitivity diagram in Expert Choice as shown in Figure 2. The x axis represents priority weights for portability. The y axis represents priority weights for scalability.

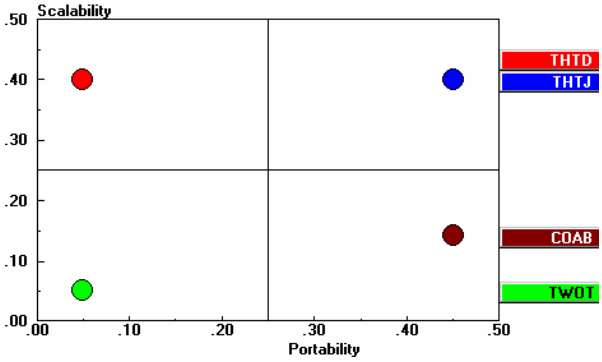

Figure 2. Tradeoff Diagram between Portability and Scalability

Figure 2 is useful since now we can visualize the tradeoffs, their relative sizes, and relationships between design alternatives in terms of tradeoffs.

For each individual architecture alternative, a point on the figure could represent a tradeoff, as long as it does not appear right on the 45 degree diagonal line of the pane. If we do not give consideration to their sizes, these tradeoffs are too numerous to be useful. For multiple architecture alternatives, if they are on the same horizontal or vertical lines, there is no tradeoff being made regarding the quality attribute while selecting any of the alternatives.

In our approach, we can divide the area in Figure 2 into four quadrants. This divides the relative size of the tradeoff into four groups. Design alternatives falling into the upper left and bottom right quadrant indicate the relatively important tradeoffs being made if the design alternative is chosen. In Figure 2, for a THTD decision (upper left corner), an architect has made a tradeoff in terms of preferring scalability to portability, while a COAB decision favors portability over scalability. The size of the tradeoff is indicated by the extent a point is positioned towards the upper left or bottom right corner. The closer it is positioned to the corner, the larger the magnitude of the tradeoff being made. Design alternatives falling into the bottom left quadrant indicate both quality attributes are negatively affected. Design alternatives falling into the upper right indicate both quality attributes are positively affected.

We produced a total of 21 tradeoff diagrams (from 7 quality attributes pair-wise) with 36 important tradeoffs identified. With these, architects have the opportunity to inspect trade-offs based on their relative sizes and

positions. The results for a selected design alternative can be documented as part of the design rationales and consequences for future evaluation and evolution in subsequent development phases.

4.1.2 Tradeoffs with quality attribute weights

In the previous analysis, tradeoffs between any two quality attributes did not include the relative weights of the quality attribute. These weights sometimes should be considered as key factors for the magnitude of the tradeoffs. When we include weightings in the analysis, some of the original tradeoffs we treated as important in the last section between two low weight quality attributes could become less significant in light of a high priority weight quality attribute.

In order to represent this characteristic of the data, we multiply each row of Table 1 with the corresponding weight of the quality attribute (available from left side of Figure 1), and obtain Table 2.

Table 2. Design Alternatives Ranking Weighted with Quality Attribute Priority

Quality Attributes	Design Alternatives			
	THTJ	THTD	TWOT	COAB
Modifiability	0.1459	0.0510	0.0129	0.0700
Scalability	0.0330	0.0330	0.0044	0.0117
Performance	0.0198	0.0198	0.0337	0.0239
Cost	0.0224	0.0162	0.0657	0.0306
Dev. Effort	0.0205	0.0149	0.0695	0.0301
Portability	0.0423	0.0047	0.0047	0.0423
Ease of Inst.	0.0297	0.0651	0.0453	0.0368

The same tradeoff diagram in Figure 2 now becomes Figure 3. This is because modifiability is the dominating quality attribute in this project. All tradeoffs related to other quality attributes are dwarfed by tradeoffs related to modifiability as we see in Figure 3.

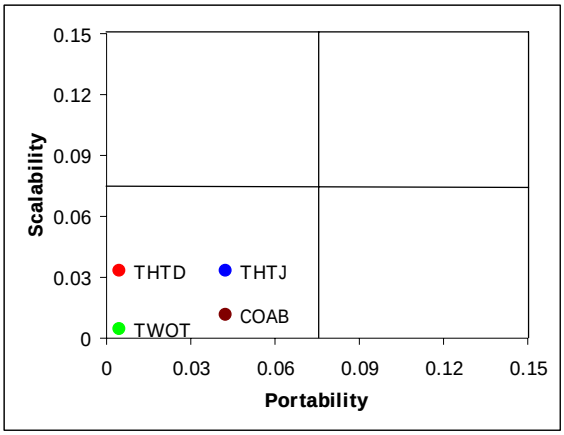

Figure 3. Tradeoff Diagram between Portability and Scalability (Weighted with Quality Attribute Priority)

The THTJ alternative has a significant advantage in modifiability among all four design alternatives (row "modifiability" in Table 1). So for THTJ, the tradeoffs between modifiability and any other quality attributes are amplified in a tradeoff design diagram as we have shown in Figure 4. THTJ only has an important tradeoff between modifiability and performance.

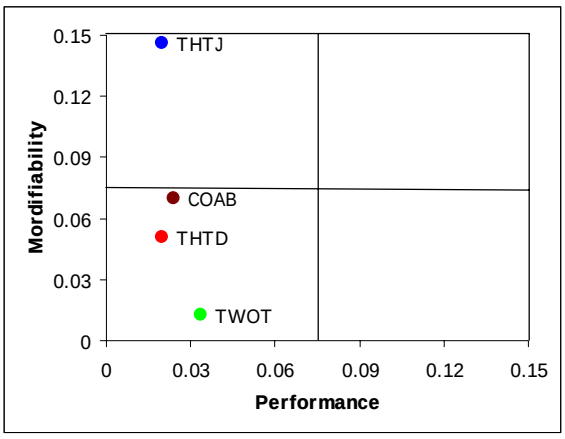

Figure 4. Tradeoff Diagram between Performance and Modifiability (Weighted with Quality Attribute Priority)

These new tradeoff diagrams with quality attribute weights considered can be used to complement the tradeoff diagrams in the last section. They help stakeholders to focus on the project-wide important tradeoffs.

Hence, these new diagrams visualize the tradeoffs and their sizes that are implicit in AHP. The process to produce these visualizations also identifies the most important tradeoffs among numerous potential candidates. The list is finally narrowed down to a small number of the most important tradeoffs based on the relative weights of the quality attributes. All these are valuable additional outputs beyond those produced by AHP.

4.2 Sensitivity Analysis

In an architecture evaluation, the quality attribute with the highest priority weight (as modifiability in Figure 1) is usually considered the most important and critical criteria. One may intuitively think that any judgments related to this quality attribute will be the most critical and may affect the final ranking. However, this is not the case in AHP. To illustrate this, here we present a simple explanation of AHP concepts in the architecture evaluation domain. More details can be found in (Saaty 1980).

Let us have M design decisions and N quality attributes. Alternatives are denoted as A_i (for i=1,2,3…,M) and criteria Q_j (for j = 1,2,3, …,N). We assume for each quality attribute Q_j, the decision maker will determine its weight W_j through pair-wise comparison. In AHP,

$$\sum_{j=1}^{N} W_j = 1$$

Let $D_{k,i,j}$ ($1 \le i < j \le M$ and $1 \le k \le N$) denote the minimum change, in absolute terms, in weight W_k of quality attribute Q_k such that the ranking of alternative A_i and A_j will be reversed. When AHP is used, $D_{k,i,j}$ can be calculated as follows (Triantaphyllou, Kovalerchuk et al. 1997):

$$D'_{k,i,j} = \frac{\left|(P_j - P_i)\right|}{\left|(a_{jk} - a_{ik})\right|} \times \frac{100}{W_k} \quad (1)$$

P_j and P_i respectively denote the final preference value of design alternative A_j and A_i. a_{jk} and a_{ik} denote relative importance of alternative A_j and A_i in terms of criteria k.

The most sensitive and critical decision can be determined by finding out the smallest number for all D values. If $D_{k,i,j}$ is the smallest number, it can have two implications:

1) The weight for quality attribute Q_k is the most sensitive and critical decision among priority weights for quality attributes.

2) The relative weight of design alternative A_j and A_i in terms of Q_k are the most sensitive and critical decisions

As we will demonstrate, it is not always the case that Q_k has the highest current W_k.

One point should be noted here. As demonstrated, we are not only interested in the best choice but also other possible order changes. In reality, due to unforeseen factors beyond the quality attributes that are not modeled in the AHP model, the first choice candidate could be dismissed in favor of a second choice or even a third one. Moreover, given that it is possible that priorities may change over time, a second choice may end up being the best choice under new priority weightings. Thus, we consider multiple rankings to be important.

Applying Eq. (1) to our data, we obtained all the values of D for a possible ranking change. For each quality attribute, there could be multiple D values that could cause a final ranking change. We only present the smallest D for each quality attribute in Table 3. The number in the final column is a percentage. As we can see, the numbers are relatively small and hence difficult to anticipate in normal decision making and requirement negotiation.

Table 3. Smallest Change in Quality Attribute Weight to Alter a Ranking

Quality Attributes	Design Alternative *i*	Design Alternative *j*	Smallest Change
Performance	TWOT	COAB	9.4
Cost	TWOT	COAB	5.1
Dev. Effort	TWOT	COAB	3.1
Portability	TWOT	COAB	2.4
Ease of Inst.	TWOT	COAB	13.5
Scalability	COAB	THTD	5.7
Modifiability	COAB	TWOT	3.9

As can be seen from Table 3 , although it may be seen as counter-intuitive, portability has the second lowest weight among all quality attributes and it is the most sensitive to change. Out of 100, a slight change of 2.4 in its weight may cause a different ranking of the final architecture choice between the TWOT and COAB design alternatives. This shows that the sensitive points have little to do with the real significance of the quality attributes.

In addition to the weight assigned to portability, a change in relative weights of TWOT and COAB in terms of portability could also cause a ranking change. Since this weight is determined by pair-wise comparison, possible inaccurate judgments could be caused by multiple sources. Most AHP tools allow you to inspect the weight of design alternatives and alter them directly to ensure rank-order consistency. For further information on ensuring rank-order consistency by adjusting pair-wise comparison, refer to (Finan and Hurley 1996).

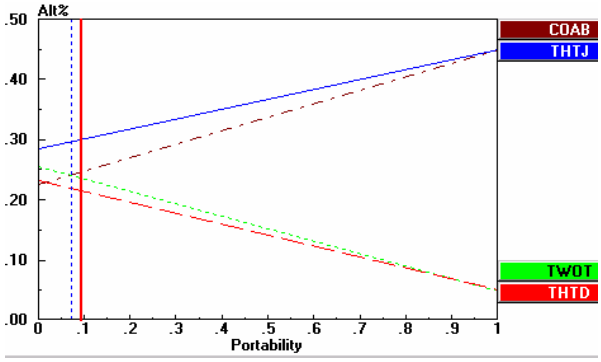

Figure 5. Gradient Diagram for Portability

In Expert Choice, D can also be located by utilizing gradient diagrams. Figure 5 is a gradient diagram for portability. The vertical line represents the priority weight of portability and is read from the x axis intersection. The priorities for the design alternatives are read from the y axis. They are determined by the intersection of the alternative's line with the quality attribute (vertical) priority line. As the vertical line moves along the x axis by changing its priority weight, the intersections with the horizontal lines represents the new priority of the design alternative read from y axis. When the vertical line meets an intersection of two design alternatives, the final ranking of the two alternatives will be altered. So the D value for two design alternative in terms of a quality attribute is the distance between the current vertical line and the intersection of two design alternatives. For Figure 5, the D value for COAB and TWOT is the distance between the vertical line and the dotted vertical line which denotes the intersection. If architects and stakeholders are not interested in the most sensitive points, they can examine the diagrams visually to inspect any intersections which are relatively close to the current priority.

If two lines denoting design decisions never cross, this means that no matter how the quality attributes are weighted, the ranking of the design alternative will never change. As shown in Figure 5, lines representing THTJ and THTD never meet, which means that no matter how portability is weighted, the rankings of THTJ and THTD will never change.

In industrial projects, the priority of quality requirements will change as business requirements change. Stakeholders may therefore speculate on possible changes of weights for certain quality attributes and use

sensitivity analysis to evaluate architecture alternatives in the new priority setting. In order to accommodate future changes, an alternative architecture in lieu of the first choice can be selected. The smallest D for each quality attribute as shown in Figure 5 can be an indicator of architecture sensitivity in terms of each quality attribute. In an architecture evaluation, all these results should be documented as design rationales and explicit consequences of the design choice.

Hence, this method makes it possible to analyze the sensitivity of an AHP based architecture ranking. The smallest change which can alter a ranking is usually not related to the quality attribute which has the highest weight. The sensitivity measurement can be both calculated through formulas and visualized in gradient diagrams. The motivation of our sensitivity analysis is to identify such subtle but critical intermediate decisions. The results can be used to:

1) Expose critical decisions which stakeholders and architects should be aware. These decisions include both priority weights of the quality attributes and priority weighs of design alternatives related to those quality attributes. Some of these might not reflect the genuine judgment of architects and stakeholders.

2) Help to identify the capabilities of the architecture to accommodate future priority changes and quantify the associated decision sensitivity

5. Discussion

5.1 Decision making process and tradeoff analysis

As we have noted, architecture evaluation can be seen as a phase of a decision making process. While we are focusing on the final stages (Selections and Evaluations) of a decision-making process in this paper, it is important to understand the intermediate decisions leading up to the final decision. Jennings and Wattam (Jennings and Wattam 1998) have discussed two approaches in making decisions: the branch approach and the root approach. In the branch approach, the Decision Maker (DM) starts with limited alternatives and tries to improve those alternatives addressing different goals. The ATAM largely follows this approach by starting with one architecture, identifying tradeoffs and improving the architecture iteratively. It is an incremental decision-making process following small steps in which tradeoffs are more easily identified. In the root approach, the DM has an extensive knowledge of a range of alternatives that is relevant to the problem, together with a detailed knowledge of the consequence of those choices. However, the DM needs to identify the tradeoffs that can be applicable between multiple criteria (Lindblom 1959). The AHP follows this approach. The analysis technique presented in this paper helps to identify these tradeoffs in AHP. Not only are tradeoffs identified between conflicting quality attributes, but also they are understood in terms of their relative importance and position in the overall project context.

5.2 Open end decisions and critical decisions

In this paper architecture evaluation is considered to be a decision-making process that contains open-ended components. We emphasize the need to reexamine final decisions (Kleindorfer, Kunreuther et al. 1993). The following two paths are suggested to assist with what should be re-examined and how.

1) Examine factors that are not considered in the current decision-making process - for example cost, time constraints, political issues, and so on.. Decisions can be also re-examined against the factors' absolute values, even though factors have been already considered as relative values in the current decision process. In (Al-Naeem, Gorton et al. 2005), cost has its own priority weight modeled in AHP, but it is re-examined as a global constraint later by using IP with absolute values. This process could be also be done in terms of benefit-to-cost ratios for each alternative as shown in (Kleindorfer, Kunreuther et al. 1993).

2) Examine the most critical decisions

Any final design decisions can hinge on sensitive and critical design decisions which may not originally be considered important. Traditionally, these critical decisions are considered to be associated with the most important criteria. We have shown this is not the case in AHP and proposed methods to discover the more subtle cases. Subsequently they can be examined as to whether they reflect the genuine thoughts of stakeholders and architects.

5.3 Requirements volatility and uncertainty in decision making

Sometimes, the first choice of current ranking may not be the best choice. This might be caused by stakeholder expectations for future quality-attribute change or the uncertainty of the current decision.

1) Changing requirements

Requirements will change with time, and so will the quality attributes and their priorities. The impacts of high volatile requirements on software development can not be underestimated (Nurmuliani, Zowghi et al. 2004). Standard requirements volatility is defined as the ratio of the number of requirements change (i.e. addition, deletion, and modification) to the total number of requirements for a certain period of time. However, not only the number of changes matters, the magnitude of each change also matters. In this paper, we measure the magnitude of quality requirements change using priority. We provide the precise priority limit for each quality attribute. This limit indicates the range of quality requirements change which the current decision can support.

2) Uncertainty of the decision.

When an open-ended decision is produced by architecture evaluation, the uncertainties associated with the decision must be measured. There can be different ways to achieve this (Kleindorfer, Kunreuther et al. 1993). In (Svahnberg, Wohlin et al. 2003), FVS on the final ranking can be one general measure of decisions uncertainty. The technique presented in this paper reveals more valuable information in terms of uncertainty for the final choice in the context of priority change.

5.4 Development life cycle and practical applicability

Within the life cycle of software development, architecture evaluation can be conducted at different stages. An early architecture evaluation can significantly reduce risks. A late life cycle architecture evaluation can be more accurate since more detailed data is available. Ever changing business requirements, implementation deviation and evolution deterioration also require architecture to be frequently evaluated. In any of such evaluation, an initial evaluation can be conducted using methods like ATAM. If the business benefits are not the immediate concerns of the evaluation and a quantitative multi-criteria decision regarding the overall architecture quality needs to be made, AHP could be used and our approach can be considered as a complement to the

existing AHP methods. We have stressed our approaches' goal on dealing with changing quality priority. This advantage along with the tool support enables our technique to be used in an iterated development process throughout the development life cycle.

Structured architecture evaluation methods have yet to be taken up by the industry. Some methods like ATAM have been validated in industry. But most of the architecture evaluation methods have not been systematically tested including AHP related ones. Our approach is based on standard AHP practice. The additional analysis introduced is easy to understand if the users are already using AHP. The current analysis is fully supported by the tool so the hurdle between a standard AHP and our additional analysis is minimal. But we realize the practical applicability of our approach depends on the overall introducing of structure methods like ATAM, ABAM and AHP into architecture evaluation.

On the other hand, empirical studies need to be conducted in industry settings to further investigate the cost and benefits of the methods. One of the limitations of our approach is that it has not been further validated except the case study conducted by the researchers so far.

6. Conclusion and Future work

In this paper, we have made the following contributions in the context of AHP:

Methods like AHP take a holistic view of the design and hide tradeoffs behind the data or provide an exponential number of tradeoffs through pair-wise comparisons. These are not helpful in understanding the design and making the decision-making process more transparent. In this paper, we provide a way to identify all possible tradeoffs and the magnitude of tradeoffs in both a local and a project-wide context based on the relative weights of a quality attribute. The results enrich the outputs of an AHP evaluation by making design consequences explicit. To this end, we hope to integrate ATAM tradeoff analysis with AHP tradeoff analysis and provide a more unified method in tradeoff analysis.

The question of sensitivity of architecture evaluation methods was first raised in (Dobrica and Niemela 2002). A sensitivity analysis for qualitative evaluation methods requires experiments or other empirical studies to be performed. AHP, on the other hand, is a quantitative method for performing architecture evaluation. We have proposed a new analysis technique to answer this question. The most sensitive critical decisions are obtained by looking for the smallest change that will alter a final alternative ranking. The change value can be an indicator of architecture sensitivity.

Finally, sensitivity analysis provides a priority weight range for quality attributes. If future priority changes are within these ranges, the current ranking remains valid. For a weight beyond this range, another alternative might become the better choice. We intend to investigate the relationship between the absolute value of quality requirements and the weighted range to further understand the capability of an architecture in handling changing quality requirements.

7. Acknowledgments

National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. We would also like to thank all the anonymous reviewers for their insightful comments on the paper.

8. References

Al-Naeem, T., Gorton, I., Babar, M. A., Rabhi, F. and Benatallah, B. 2005. A Quality-Driven Systematic Approach for architecting Distributed Software Applications. Proceedings of the 27th International Conference on Software Engineering (ICSE), St. Louis, USA

Aurum, A. and Wohlin, C. 2003. The fundamental nature of requirements engineering activities as a decision-making process. Information and Software Technology 45(14): 945-954.

Bass, L., Clements, P. and Kazman, R. 2003. Software Architecture in Practice, Addison-Wesley.

Bengtsson, P., Lassing, N., Bosch, J. and Vliet, H. v. 2004. Architecture-level modifiability analysis (ALMA). Journal of Systems and Software 69(1-2): 129-147.

Bosch, J. 2000. Design & Use of Software Architectures: Adopting and evolving a product-line approach, Addison-Wesley.

Bosch, J. 2004. Software architecture: the next step. Proceedings of the First European Workshop on Software Architecture (EWSA), pp. 194-199

Chung, L., Nixon, B. A., Yu, E. and Mylopoluos, J. 2000. Non-Functional Requirements in Software Engineering. Dordrecht, Kluwer Academic Publishers.

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

Expert Choice (version 11). 2004. http://www.expertchoice.com/ Last accessed on 26th Sept., 2004

Finan, J. S. and Hurley, W. J. 1996. A note on a method to ensure rank-order consistency in the analytic hierarchy process. International Transactions in Operational Research 3(1): 99-103.

Giorgini, P., Mylopoulos, J., Nicchiarelli, E. and Sebastiani, R. 2004. Formal reasoning techniques for goal models. Journal on Data Semantics 1: 1-20.

Gorton, I. and Haack, J. 2004. Architecting in the Face of Uncertainty: An Experience Report. Proceedings of the 26th International Conference on Software Engineering (ICSE), Edinburgh, United Kingdom, pp. 543-551

Gross, D. and Yu, E. S. K. 2001. From Non-Functional Requirements to Design through Patterns. Requirement Engineering 6(1): 18-36.

In, H., Boehm, B., Rodgers, T. and Deutsch, M. 2001. Applying WinWin to Quality Requirements: A Case Study. Proceedings of the 23rd International Conference on Software Engineering (ICSE), pp. 555-564

In, H., Kazman, R. and Olson, D. 2001. From Requirements Negotiation to Software Architectural Decisions. Proceedings of the First International Workshop from Software Requirements to Architectures (STRAW'01)

Jennings, D. and Wattam, S. 1998. Decision making: an integrated approach. London ; Washington, D.C., Financial Times Pitman Pub.

Karlsson, J., Wohlin, C. and Regnell, B. 1998. An evaluation of methods for prioritizing software requirements. Information and Software Technology 39(14-15): 938-947.

Kazman, R., Asundi, J. and Klein, M. 2001. Quantifying the costs and Benefits of Architectural Decision. Proceedings of the 23rd International Conference on Software Engineering (ICSE), pp. 297-306

Kazman, R., Barbacci, M., Klein, M. and Carriere, J. 1999. Experience with Performing Architecture Tradeoff Analysis. Proceedings of the 21st International Conferences on Software Engineering (ICSE'99), pp. 54-63

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

Kleindorfer, P. R., Kunreuther, H. and Schoemaker, P. J. H. 1993. Decision sciences: an integrative perspective. Cambridge, England ; New York, N.Y., Cambridge University Press.

Lindblom, C. E. 1959. The Science of Muddling Through. Public Administrative Review 19: 79-88.

Liu, L. and Yu, E. 2001. From Requirements to Architectural Design - Using Goals and Scenarios. Proceedings of the First International Workshop from Software Requirements to Architectures (STRAW'01)

Nurmuliani, N., Zowghi, D. and Powell, S. 2004. Analysis of requirements volatility during software development life cycle. Proceedings of the Software Engineering Conference, 2004. Proceedings. 2004 Australian, pp. 28-37

Saaty, T. L. 1980. The Analytic Hierarchy Process. New York, McGraw Hill.
Svahnberg, M. and Wohlin, C. 2002. Consensus Building when Comparing Software Architectures. Proceedings of the 4th International Conference on Product Focused Software Process Improvement (PROFES), pp. 436-452
Svahnberg, M., Wohlin, C., Lundberg, L. and Mattsson, M. 2002. A method for understanding quality attributes in software architecture structures. Proceedings of the 14th international conference on Software engineering and knowledge engineering (SEKE), pp. 819-826
Svahnberg, M., Wohlin, C., Lundberg, L. and Mattsson, M. 2003. A Quality-Driven Decision-Support Method for Identifying Software Architecture Candidates. International Journal of Software Engineering and Knowledge Engineering 13(5): 547-573.
Triantaphyllou, E., Kovalerchuk, B., Jr, L. M. and Knapp, G. M. 1997. Determining the most important criteria in maintenance decision making. Journal of Quality in Maintenance Engineering 3(1): 16-28.
Triantaphyllou, E. and Mann, S. H. 1994. Some critical issues in making decisions with pair-wise comparisons. Proceedings of the Third International Symposium on the AHP, pp. 225-36

Index terms:
Software architecture, architecture evaluation, analytic hierarchy process, trade-off, sensitivity analysis, decision making, multi-criteria decision making, non functional requirements, quality attributes

Mailing address:

Liming Zhu

National ICT Australia
Locked Bag 9013
Australian Technology Park
Alexandria
NSW 1435
Australia

Phone: +61 2 83745523
Fax: +61 2 8374 5520
Email: limingz@cse.unsw.edu.au

Liming Zhu is a PHD candidate in the School of Computer Science and Engineering at University of New South Wales. He is also a member of the Empirical Software Engineering Group at National ICT Australia (NICTA). He obtained his BSc from Dalian University of Technology in China. After moving to Australia, he obtained his MSc in computer science from University of New South Wales. His principle research interests include software architecture evaluation and empirical software engineering.

Aybüke Aurum is a senior lecturer at the School of Information Systems, Technology and Management, University of New South Wales. She received her BSc and MSc in geological engineering, and MEngSc and PhD in computer science. She also works as a visiting researcher in National ICT, Australia (NICTA). Dr. Aurum is one of the editors of "Managing Software Engineering Knowledge", "Engineering and Managing Software Requirements" and "Value-Based Software Engineering" books. Her research interests include management of software development process, software inspection, requirements engineering, decision making and knowledge management in software development. She is on the editorial boards of Requirements Engineering Journal and Asian Academy Journal of Management.

Ian Gorton is a Senior Researcher at National ICT Australia. Until Match 2004 he was Chief Architect in Information Sciences and Engineering at the US Department of Energy's Pacific Northwest National Laboratory. Previously he has worked at Microsoft and IBM, as well as in other research positions. His interests include software architectures, particularly those for large-scale, high-performance information systems that use commercial off-the-shelf (COTS) middleware technologies. He received a PhD in Computer Science from Sheffield Hallam University.

Dr. Ross Jeffery is Professor of Software Engineering in the School of Computer Science and Engineering at UNSW and Program Leader in Empirical Software Engineering in National ICT Australia Ltd. (NICTA). His current research interests are in software engineering process and product modeling and improvement, electronic process guides and software knowledge management, software quality, software metrics, software technical and management reviews, and software resource modeling and estimation. His research has involved over fifty government and industry organizations over a period of 15 years and has been funded from industry, government and universities. He has co-authored four books and over one hundred and twenty research papers. He has served on the editorial board of the IEEE Transactions on Software Engineering, and the Wiley International Series in Information Systems and he is Associate Editor of the Journal of Empirical Software Engineering. He is a founding member of the International Software Engineering Research Network (ISERN). He was elected Fellow of the Australian Computer Society for his contribution to software engineering research.