UNIVERSITY OF GOTHENBURG

A method of selecting appropriate software architecture styles: Quality Attributes and Analytic Hierarchy Process

Bachelor of Science Thesis in the Programme Software Engineering&Management

Qiushi Wang

Zhao Yang

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2012

A method of selecting appropriate software architecture style:
Quality Attributes and Analytic Hierarchy Process

Qiushi Wang
Zhao Yang

Examiner: Helena Holmström Olsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover picture taken from: http://blog.art21.org/wp-content/uploads/2009/11/Malibu-5-Stephen-Kanner.jpg

Department of Computer Science and Engineering
Göteborg, Sweden, May 2012

A method of selecting appropriate software architecture style/pattern: Quality Attributes & Analytic Hierarchy Process

Qiushi Wang
Dept.Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
Email: guswangqi@student.gu.se

Zhao Yang
Dept. Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
Email: joeyzhaozhao@gmail.com

Abstract - Software Architecture Style is a proven reusable solution for known problems that in order to save huge cost and reduce risks. Software development can benefit from correct architecture style. Thus, architecture style selection is important when design software system. In this research, the authors devote to create a selection method for people who lack expertise and experience to select appropriate architecture style for their software systems. The authors collect and categorize a number of common architecture styles, and use Quality Attributes as a criterion to evaluate all those architecture styles. Moreover, they provide a systematic selection process powered by Analytic Hierarchy Process (AHP).

Keywords - software architecture style; software architecture selection; quality attributes; analytic hierarchy process.

1. INTRODUCTION

Software Architecture is a rising subject of software engineering to help people to oversee a system in high level (Qin et al., 2007, p.1). It is a critical aspect in the design and development of software (Vijayalakshmi et al., 2010). Software Architecture involves a series of decisions based on many factors in a wide range of software development, and each of these decisions can have considerable impact on the overall success of the software (Microsoft, 2009, p.3).

Good software architecture can reduce the business risks associated with building a technical solution (Microsoft, 2012, p.5), and make the system implementation and testing more traceable as well as achieve higher Quality Attributes (Qian et al., 2008, p.2). It paves the way for software success (Northrop, 2003). On the contrary, poor software architecture makes software production inefficient in terms of cost and time (Qian et al., 2008, p.2), and it usually can lead to disaster (Northrop, 2003).

Software architects are the people who take responsibility to develop the architecture design (Qian et al., 2008, p.2) and their most important job is to map software requirements to architecture design and guarantee that both functional requirements and Quality Attributes are met (Qin et al., 2007, p.4). Architects might face similar issues in different software architecture design, and some of those issues are not new. For saving of huge cost and the reduction of risks, software architecture can be reused (Qin et al., 2007, p.1).

Software architecture style (also known as “architecture pattern”) is a proven reusable solution for known problems and it is built on tried and tested design experience (Buschmann et al., 2007). Qian et al (2007, p.8) states that an architecture style contains a set of rules, constraints and patterns of how to structure of a system into a set of elements and connectors. In most cases, a software system has its own application domain, each domain has its own reference model and an architecture style is a viewpoint abstraction for a software structure that is domain-independent (Qian et al., 2008, p.9).

An appropriate architecture style can improve partitioning and promotes design reuse by providing solutions to frequently recurring problems (Microsoft, 2009, p.20). With the development of software architecture design, a number of architecture styles are created and used/reused to address various of problems. Every architecture styles has its own history and certain context, in other words, each architecture style is proposed in a certain environment and can solve certain key problems or satisfy certain requirements (Qin et al., 2007, p.35).

As we know an architecture style that is proper for all systems does not exist because systems have different requirements (Qin et al., 2007, p.35), and as mentioned above, system can benefit from architecture style (only)

when appropriate style is selected. Thus, the choice of which architecture to go with is an important part in any software development because this choice affects the quality of the final software product (Vijayalakshmi et al., 2010). So selecting an appropriate style for a system is a question that should be brought up when architects design the software architecture.

The major focus of our thesis work is about how to select appropriate style for software system; based on this main problem we pose such two research questions:

RQ1: What software architecture styles are commonly used today?

RQ2: How to select proper architecture style?

In order to answer the two research questions we mentioned above, we are going to gather a number of commonly used architecture styles at present and categorise them based on their scope of application, and then we give a criterion of evaluating/comparing architecture styles as well as a scientific method to select the most appropriate one. And in order to help audiences to understand our selection method we conduct a case study of a web-based business to business (b2b) system as an instruction of applying our selection method.

With the increasing complexity of software systems, multiple architecture styles are usually utilized in the same project (Qian et al., 2008, p.266). Our selection method and criteria can be used to find appropriate styles for single software system as well as the subsystem of complex/large software systems.

Several similar research articles were found by us in our literature study, and by comparison with those researches, our category of architecture styles that is based on scope of application can offer relatively accurate candidates styles; and the computing of decision-making process is less complicated. In a word, it can be an effective method with ease of use.

We introduce related work in section 2 including Quality Attributes and a decision-making model named Analytic Hierarchy Process. Research approaches are described in section 3, and the collected data is analysed and displayed in section 4. The case study is in section 5, and in section 6 we recapitulate major findings and position our contribution. Finally, we make our conclusion as well as our suggestion to future work in section 7.

2. RELATED WORK

This section explains two important concepts related to our research, it helps readers to understand and apply the method we created.

2.1 Quality Attributes

Software architecture is typically specified in different views to show the relevant functional and non-functional requirements (also known as Quality Attributes) of a software system (Buschmann et al., 2001). Functional requirements deal with a particular aspect of a system's functionality, and are usually related to a specified functional requirement such as particular function and compute algorithm (Buschmann et al., 2001). On the contrary, Quality Attributes are the overall factors that affect run-time behaviour, system design, and user experience (Meier et al., 2009). They represent features of a system that functional requirements do not cover and typically addresses aspects related to the reliability, compatibility, cost, ease of use, maintenance or development of a software system (Buschmann et al., 2001). The desired combination of Quality Attributes indicates the success of the design and quality of the system. When designing a software application, it is not enough to merely satisfy functional requirements; fulfilling the Quality Attributes is also required. It is necessary to analyse the tradeoffs between multiple Quality Attributes since the priority of each Quality Attributes differs from system to system, and has exist the potential to impact on other requirements as well (Meier et al., 2009).

2.2 Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process (AHP) is a mathematical decision-making technique that proposed by Saaty (1980). The AHP deals with problems of how to measure intangible criteria and how to interpret correctly measurements of tangibles; so they can be combined with those of intangibles to yield sensible, not arbitrary numerical results (Satty, 2005). It is a widely used theory and provides a measurement through pairwise comparisons and relies on the judgments of experts to derive priority scales (Saaty, 2008).

In order to apply AHP in an organized way to generate priorities; it needs to break down the decision into a few steps:

- Define the problems and determine the related knowledge.
- Structure the decision hierarchy model from the top with the goal of the decision through the intermediate levels to the lowest level (a set of the alternatives).
- Construct a set of pairwise comparison matrices.
- Weigh the priorities for every element by using the priorities obtained from the comparisons.

The mathematics of the AHP and the calculation techniques are briefly explained in the following. Initially, assigning a number to each element on a scale that indicates how many times more important one element is over another element. The rating scale adapted from Saatys' fundamental scale of absolute numbers. These pairwise comparisons are carried out for all factors to be considered; after that, the matrix is completed. The next step is to calculate a consistency ratio (CR) to measure how consistent comparisons are. If the CR is less than 0.1, that indicates good consistency. The third step is to calculate the list of elements' priority vectors, which express the relative weight of each element type. The final stage is to compute the total score by adding the score of elements' priority values and the results with the highest total score is chosen (Coyle, 2004; Saaty, 2005; Saaty, 2008; Galster et al., 2010).

3. REASEARCH APPOARCH

This research is conducted by two researchers with education background of Software Engineering & Management.

As mentioned in the Introduction, the core of our thesis work is to create a method for selecting appropriate software architecture style. In order to methodically conduct the research, we decide to breakdown the research problem into smaller sequential parts. The benefit of breakdown is that each smaller part has a clear objective and characteristic, and we can easily make changes on specific parts. We divide our research into five steps:

a. Collecting a number of commonly used software architecture styles at present.

b. Categorization of collected styles based on their scope of application.

c. Research on how to select architecture style as well as evaluation of software architecture and then decide selection criteria and method.

d. Analysis on collected architecture styles based on our criteria.

e. Create a selection process.

3.1 Literature Study

Literature review is our major approach for data collection in this research. Our literature review aims at three specific aspects: commonly used software architecture styles at present, methods of selecting/comparing architecture styles and research/analysis articles on specific architecture styles.

There are two major sources of literature for us to seek information: published books and articles published in well-known electronic databases, listed below:

- **IEEEXPLORE** http://ieeexplore.ieee.org
- **ACM Digital Library** http://dl.acm.org/
- **Chalmers Library** http://www.lib.chalmers.se/
- **SCIRUS** http://scirus.com/
- **SpringerLink** http://www.springerlink.com

3.1.1 Commonly used architecture styles

We decided to look for published books what systematically elaborate software architecture and architecture styles. After this step, we should give a list of categorized (based on scope of application, e.g. Web Service, Distributed system) software architecture styles.

Qian et al.'s (2008) book *Software Architecture and Design Illuminated* provides a coherent and integrated approach to the discipline of software architecture design. The book also covers a complete set of important software design methodologies and architecture styles as well as details of these architecture styles.

Qin et al.'s (2007) book *Software Architecture* provides introduction to the theory foundations, various sub-fields, current research status and practical methods of software architecture. It can be used as a learning material for accessing software architecture. In this book, readers can acquire the basic knowledge of software architecture, including what architecture styles are popular for practice use and how we can apply software architecture into the development of systems; the information about popular architecture styles is quite valuable for us.

Zhu Hong's (2005) book *Software Design Methodology: from principles to architectural styles* is based on the author's lecture notes prepared for teaching a Software Design module at Oxford Brookes University to software engineering students over 6 years. In one section he introduce and analyse 5 groups of typical architecture styles.

We also found a book named *Microsoft Application Architecture Guide* (Microsoft, 2009). It is a Microsoft Press book available on the MSDN library, it provides guidance for using architecture principles, design principles, and software architecture patterns/styles that are tried and trusted.

The four books gather a number of main architecture styles as well as their strengths, limitations, and applicable domains that is helpful for us to conduct analysis on these styles later. The method of categorizing architecture styles is that we extract helpful information from four books and list all architecture styles, and then we discuss to decide what category each of them belong.

3.1.2 Criteria and methods of selecting/comparing architecture styles

In this part we seek articles about selecting or comparing software architecture styles or evaluation of software architecture. After reviewing selected articles, we should conclude the factors for comparing and evaluating architecture styles as criteria for selection. Besides, we are trying to find a systematic method for decision-making from found articles.

Search Terms:

We defined a number of keywords for the search engines of the electronic database. The following five different combinations of keywords returned significant results:

- software architecture styles
- evaluating software architecture
- software architecture selection criteria
- software architecture analysis
- comparing architecture styles
- selecting software architecture style

Included criteria:

- The articles refer to theoretical concepts in context of software architecture domains
- The articles illustrate software architecture styles selection criteria
- The articles provide an methodologies of evaluating different architectural styles
- Chapter of published books

Number of relevant articles	After applying included criteria	Actually used articles
56	29	11

Table 3.1 Literature on criteria of selecting architecture style.

3.1.3 Research on specific architecture styles

We get a criterion in the last step, and then we need to conduct analysis on collected architecture styles and measure each of them. After this step, all collected architecture styles are measured and marked so that we can horizontally compare each candidate's style and make a decision.

Four books mentioned in section 3.1.1 not only list commonly used architecture styles but also provide a few analysis, so we treat the four books as the major data source in this step.

3.2 Data Analysis

In the data analysis, the outcome of each data collection step will be integrated as a whole to be analysed. The data from each collection step will affect each other and cause modification and elimination in order to improve reliability of our research.

4. RESULT

This section presents the collected data by conducting our research approaches. All data displayed here have been analysed by the two authors.

4.1 Commonly used architecture styles in category

We extract information from the four books listed in section 3.1.1 and combine the data; and then we eliminate architecture styles what are relatively uncommon, for instance, some styles are mentioned in only one book. Moreover, for some architecture styles that we could not find ample information to support their applicable domain, benefit and limitation, we eliminate them as well in order to improve reliability of this thesis. All collected architecture styles are represented in Table 4.2 (in page 8).

4.2 Selection method

The selection method includes two parts: evaluation criteria and selection process. The evaluation criteria are factors we used to measure each architecture style in order to compare them horizontally; the selection process is a designed series of steps to find out the most appropriate style. We recommend using Quality Attributes as evaluation criteria and Analytic Hierarchy Process (AHP) for selection process.

4.2.1 Quality Attributes

In order to select correct architecture style, we should consider different aspects that related to this objective, for instance, functional requirements and nonfunctional requirements (also known as Quality Attributes), architect's priorities and the system domain, thus, choosing architecture styles has been defined as a multi criteria decision-making problem (Moaven et al., 2008a; Babu et al., 2010; Vijayalakshmi et al., 2010). Due to the limitation of resource, it is difficult to take into account

all criteria related to problem at once in our research, therefore we attempt to find the critical element.

In recent years, a number of studies (Svahnberg et al., 2002; Moaven et al., 2008a; Moaven et al., 2008b; Babu et al., 2010; Vijayalakshmi et al., 2010) proposed that satisfying Quality Attributes is propounded as a key element in design or selection of appropriate architecture for systems. Hence, we determine Quality Attributes as the criterion of measuring architecture styles.

Quality attributes can be categorized based on their nature, effect and context. Both Qian et al (2008, p9) and Microsoft Application Architecture Guide (2009, p.192) conclude a number of common Quality Attributes in groups, however their own listed Quality Attributes and categories are not the same, hence we combine the data and create a table with all mentioned Quality Attributes what are recategorized after group discussion. We eliminate relatively uncommon Quality Attributes for reliability of evaluation.

Category	Quality Attributes	Description
Implementation attributes (not observable at runtime)	Maintainability	The ability to modify the system and conveniently the system and conveniently extend it.
	Testability	The degree to which the system facilitates the establishment of test cases. Testability usually requires a complete set of documentation accompanied by system design and implementation
	Portability	The system's level of independence on software and hardware platforms.
	Flexibility	The ease of system modification to cater to different environments or problems for which the system was not originally designed.
	Reusability	Reusability defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time.
	Simplicity	Those attributes of the software products that provide maintenance and implementation of the functions in the most understandable manner.
Runtime attributes (observable at run time)	Availability	Availability defines the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period. Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load.
	Security	A system's security's to cope with malicious attacks from outside or inside the system.
	Performance	Increasing a system's efficiency with regard to response time, throughput, and resource utilization, attributes which usually conflict with each other.
	Concurrency	Concurrency is a property of systems in which several computations are executing simultaneously, and potentially interacting with each other. The computations may be executing on multiple cores in the same chip, preemptively time-shared threads on the same processor, or executed on physically separated processors.
	Reliability	The failure frequency, the accuracy of output results, the Mean-Time-to-Failure, the ability to recover from failure, and the failure predictability.
	Scalability	A system's ability to adapt to an increase in user requests
Business attributes	Cost	The expense of building, maintaining, and operating the system.
	Lifetime	The period of time that the product is alive before retirement.
User attributes	Usability	The level of human satisfaction from using the system. Usability include matters of completeness, correctness, compatibility, as well as friendly UI, complete documentation, and technical support.
System attributes	Supportability	It refers to the ability of technical support personnel to install, configure, and monitor computer products, identify exceptions or faults, debug or isolate faults to root cause analysis, and provide hardware or software maintenance in pursuit of solving a problem and restoring the product into service. Incorporating serviceability facilitating features typically results in more efficient product maintenance and reduces operational costs and maintains business continuity.

Table 4.1 Common Quality Attributes with brief description (Qian et al., 2008; Microsoft Application Architecture Guide, 2009).

Table 4.1 includes most common Quality Attributes. We strongly recommend that when architects and key stakeholders prioritise Quality Attributes for their software system, they should use Quality Attributes listed in Table 4.1 as candidates because we measure each architecture style with all these Quality Attributes in following section. If a system with Quality Attributes that do not exist in Table 4.1, our selection method would be hard to give accurate result.

4.2.2 Architecture Styles Evaluation

As we mentioned before, we have categorized all collected architecture styles based on their scope of application, so a system can get a number of candidate's styles dependent on its nature, and then we should compare candidates to make a correct decision. In the evaluation process we measure each architecture style with all Quality Attributes listed in Table 4.1.The evaluation results are displayed in Table 4.2. We replicate the method of measuring architecture styles utilized by Galster et al., (2010). "++" represent a architecture style perform very well with some specific Quality Attribute; "+" stands for some support; "–" indicates that the style has negative impact on some specific Quality Attributes; "--" indicates that very negative impact on a Quality Attributes; "o" means no support, neutral or unsure.

STYLES \ QAs		Maintainability	Testability	Portability	Flexibility	Reusability	Simplicity	Availability	Security	Performance	Concurrency	Reliability	Scalability	Cost	Life Time	Usability	Supportability
Data Flow System	Batch Sequential	o	++	o	o	+	+	o	o	o	–	o	o	o	o	- -	o
	Pipe & Filter	+	+	o	+	+	+	o	o	o	++	o	+	o	o	- -	–
	Process control	o	o	+	o	o	o	o	o	o	o	o	o	+	o	o	o
Centralized Data Store System	Repository Arch	–	o	o	–	+	o	+	o	o	o	–	+	–	o	–	o
	Blackboard Arch	–	–	–	o	+	o	o	o	+	++	o	+	o	o	o	o
Large/ Complex System	Repository Arch	–	o	o	–	o	o	+	o	o	o	–	+	–	o	-	o
	Blackboard Arch	–	–	–	o	+	o	o	o	+	++	o	+	o	o	o	o
	Main-subroutine	–	o	o	o	–	o	o	-	o	o	+	o	o	o	o	o
	Master-slave	o	o	o	o	o	o	o	o	o	+	++	o	o	o	o	o
	Layered Arch	++	+	++	+	+	–	+	o	- -	–	o	+	o	o	o	o
Web Service	Service-Oriented	+	o	o	o	++	–	+	o	o	o	o	+	+	o	o	o
	MVC	+	o	o	+	o	+	+	o	o	o	o	- -	o	o	+	+
Distributed System	Client Server	–	–	o	o	+	+	–	++	–	o	–	+	o	o	o	o
	Broker Arch	++	–	+	+	+	+	o	–	–	o	o	o	o	o	o	o
	Service Oriented	+	o	o	o	++	–	+	o	o	o	o	+	+	o	o	o
User-Interaction Oriented System	MVC	+	o	o	+	o	+	+	o	o	o	o	- -	o	o	+	+
	Presentation Abstraction Control (PAC)	+	o	+	+	+	–	+	o	–	+	o	o	o	o	+	o

Table 4.2 Commonly used architecture styles in category and evaluation (Qian et al., 2008; Microsoft Application Architecture Guide, 2009 ; Qin et al., 2007; Zhu, 2005)

4.2.3 Selection Process

This section displays and explains the selection method we designed.

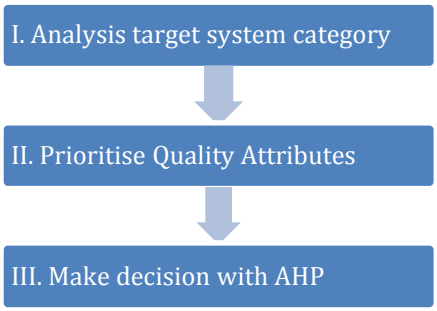

Figure 4.1 Workflow of Selection process

I. Analysis target system category

As shown in Table 4.2, all architecture styles have been categorized with their applicable domain. In this step, we should determine which group the target system belongs with and then we can get several candidates styles.

II. Prioritise Quality Attributes

In this step, architect and key stakeholders should prioritise a number of Quality Attributes what they hope the system could achieve.

III. Make Decision with AHP

We have measured and marked each architecture style in previous steps, each style is marked at 5 different levels with every Quality Attributes. In order to improve reliability of the selection, we need a systematic decision making model to support. When facing similar problem, Galster et al. (2010) utilized AHP model to solve it in mathematic way. The architecture styles selection process based on the AHP model consists of a number of architecture styles that are evaluated in terms of multiple Quality Attributes. The main steps are summarized as following.

(i) Pair-wise comparison of each element and estimation of relative importance

The determination of pair-wise comparisons between Quality Attributes has to be performed by various stakeholders (architects, domain experts, programmers etc.). These comparisons are conducted based on the rules prescribed by AHP and using Satty's (2008) fundamental scale (from 1 to 9) to measure the relative importance of each element.(in Table 4.3)

Intensity of Importance	Definition	Explanation
1	Equal Importance	Two activities contribute equally to the objective
2	Weak or slight	Experience and judgement slightly favour one activity over another
3	Moderate importance	
4	Moderate plus	Experience and judgement strongly favour one activity over another
5	Strong importance	
6	Strong plus	An activity is favoured very strongly over another, its dominance demonstrated in practice
7	Very strong or demonstrated importance	
8	Very, very strong	The evidence favouring one activity over another is of the highest possible order of affirmation
9	Extreme importance	

Table 4.3 The fundamental scale of absolute numbers (Adapted from Saaty, 2008)

(ii) Construction of the weighted matrix

The Quality Attributes are denoted by C_j (j = 1,2…n). Each Quality Attribute is associated with a scale of absolute numbers. The initial matrix A for the pair-wise comparison is presented below. In a matrix, for instance, when comparing two Quality Attributes C_1/C_2, a value of 1 is assigned if C_1 is equally important as C_2, if C_1 is absolutely more important than C_2, it should be rated at 9; conversely, the C_2 is valued at 1/9.

$$A = \begin{bmatrix} C_1/C_1 & C_1/C_2 & C_1/C_3 & \dots & C_1/C_n \\ C_2/C_1 & 1 & C_2/C_3 & \dots & C_2/C_n \\ C_3/C_1 & C_3/C_2 & 1 & \dots & C_3/C_n \\ \dots & \dots & \dots & 1 & \dots \\ C_n/C_1 & C_n/C_2 & C_n/C_3 & \dots & C_n/C_n \end{bmatrix}$$

(iii) Calculation of the consistency of the matrix

After weighted matrix is completed, the crucial thing about measuring the consistency ratio of the matrix could be calculated by the following way:

$CI = \frac{\lambda_{max} - n}{n-1}$ $\quad CR = {}^{CI}/_{RI}$

where:

CI: the consistency index

λ_{max}*: the largest eigenvalue of matrix*

n: the order of comparison matrix

CR: the consistency ratio

RI: the random consistency index (see Table 4.4)

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.52	0.89	1.11	1.25	1.35	1.40	1.45	1.49

Table 4.4 Random Consistency Index (Adapted from Satty, 2004)

λ_{max} is the largest eigenvalue of matrix, which can be calculated through the eigenvalue calculator, we found two online available calculators:

[1]*MATRIX CALCULATOR APPLET and BLUEBIT*

The consistency ratio (CR) is calculated to measure how consistent the judgements have been relative to large samples of purely random judgements (Coyle, 2004). If the *CR* is in an excess of 0.1 the judgements are untrustworthy because they are too close to randomness and the assigned value of each Quality Attribute must be reassigned (Satty, 2004; Satty, 2005; Satty, 2008; Coyle, 2004). In general, if *CR* is less than 0.1, the judgments can be considered as good consistency (Satty, 2004; Satty, 2005; Satty, 2008).

(iv) Determination of the priority vectors

Saaty (2004, 2005) proposed the eigenvalue approach to determine the desired priority vectors. The process of deriving the priority vectors refers to Ishizaka & Lusti (2006) and Saaty (2004, 2005). The priorities are derived as follow:

Step 1. Square the pair-wise matrix

The initial matrix is given below (Figure 4.1), after the step of square; the value of the matrix is shows on Figure 4.2.

$1(a_{11})$	6	2
1/6	$1(a_{22})$	1/2
1/2	2	$1(a_{33})$

Figure 4.1 Initial matrix

$3(b_{11})$	$16(b_{12})$	$7(b_{13})$
$0.583(b_{21})$	3	1.333
$1.333(b_{31})$	7	3

Figure 4.2 Squared matrix

For instance, in figure 4.2, the value a_{22} is calculated by sum of all squared b_{22} values from initial matrix in figure 4.1, $b_{22} = 3$:

$b_{22} = a_{21} * a_{12} = 1/6 * 6 = 1;$

$b_{22} = a_{22} * a_{22} = 1 * 1 = 1;$

$b_{22} = a_{23} * a_{32} = 1/2 * 2 = 1;$

The squaring of the matrix takes the sum of all the three lines values; the result is displayed on figure 4.2.

Step 2. Sum and normalise the rows

(a) Sum of the elements of each row, use the value of squared matrix (Figure 4.2)

$r_1 = 3 + 16 + 7 = 26;$

$r_2 = 0.583 + 3 + 1.333 = 4.916;$

$r_3 = 1.333 + 7 + 3 = 11.333;$

(b) Normalisation of each row. Using each element divide the sum value of each row:

$$\begin{cases} ({}^{3}/_{26}),\ ({}^{16}/_{26}),\ ({}^{7}/_{26}) \\ ({}^{0.583}/_{4.916}),\ ({}^{3}/_{4.916}),\ ({}^{1.333}/_{4.916}) \\ ({}^{1.333}/_{11.333}),\ ({}^{7}/_{11.333}),\ ({}^{3}/_{11.333}) \end{cases}$$

==

$$\begin{cases} (0.115, & 0.615, & 0.269) \\ (0.118, & 0.610, & 0.271) \\ (0.117, & 0.617, & 0.264) \end{cases}$$

Step 3. Get the approximation of priority vectors

Calculate the mean value of each column, and then get the final result of approximate priority vector ($\vec{p}$).

$c_1 = 0.115 + 0.118 + 0.117 = 0.35;$

$c_2 = 0.615 + 0.610 + 0.617 = 1.842;$

$c_3 = 0.269 + 0.271 + 0.264 = 0.804;$

Thus, the approximate priority vector is $\vec{p}$ = (0.116, 0.614, 0.268).

[1]http://www.bluebit.gr/matrix-calculator/default.aspx
http://www.math.ubc.ca/~israel/applet/mcalc/matcalc.html

(v) Computation of the total scores for each architecture styles and suggest the appropriate style

The priority vectors and the Table 4.2 are the input to the computation of the total scores for each architecture styles. The weighted score method (WSM) are utilized to weight the priority of each Quality Attribute (Galster et al., 2010). Firstly, the discrete ordinal integer values $\chi \in [-2, 2]$ represent the symbols from Table 4.2 to numerical values based on the following (Adapted from Galster et al., 2010):

$$\chi = \begin{cases} -2 \; if \; symbol = '--' \\ -1 \; if \; symbol = '-' \\ 0 \; if \; symbol = 'o' \\ 1 \; if \; symbol = '+' \\ 2 \; if \; symbol = '++' \end{cases}$$

Secondly, each element in the priority vectors is multiplied with the respective row in the Table 4.2. For instance, the value of priority vectors for maintainability of layered architecture style is multiplied with the weight of maintainability that symbols as '++' in Table 4.2.

Thirdly, compute the total score of each candidate architecture style. Architects and key stakeholders already prioritised a number of Quality Attributes in the pervious step. The weighted scores of all Quality Attributes have been set before (see Table 4.2). The scores for each prioritised Quality Attributes are calculated in the last step, and then we sum up all the scores to obtain the total score for each architecture style by using the formula:

$$\omega_{total} = \omega_1 + \omega_2 + \omega_3 + \cdots + \omega_n$$

Finally, the architecture style with the highest total score in ω_{total} is suggested as the appropriate architecture style.

5. CASE STUDY

In the following we conduct a case study that illustrates our selection method in order to help audience understanding. It is necessary to mention that the studied case here is a hypothetical case. It exactly follows the designed workflow of selection process and provides an instruction of how to apply it to audiences. The input of selection method is a software system or subsystem within some specific application domain. The expected output is the total scores with satisfaction value of each candidate style. The architecture style with the highest total score is the suggested appropriate style.

The studied system is a web-based b2b (business to business) application. The following applies each step of selection method designed in section 4.2.3.

5.1 Analysis target system category

This b2b application is a web-based system and offers various services to customers via the internet. According to characters of b2b application the category goes to WEB SERVICE (see Table 4.2). There are two candidates' architecture styles belonging to this category: service orientated architecture (A) and MVC (B).

5.2 Prioritise Quality Attributes

From the Table 4.1, the overview of candidates Quality Attributes are described, and then we prioritise four Quality Attributes that are important to this b2b application: Usability, Maintainability, Cost and Scalability.

5.3 Make decision with AHP

(i) Construct pair-wise comparison of weighted matrix

	Usability	Maintainability	Cost	Scalability
Usability	1	2	5	3
Maintainability	1/2	1	3	2
Cost	1/5	1/3	1	1/3
Scalability	1/3	1/2	3	2

Figure 5.1 Pair-wise comparison matrix of each Quality Attribute

(ii) Calculate the consistency ratio of matrix

$CI = \frac{4.059 - 4}{3} = 0.0197,$

$CR = \frac{0.0197}{0.89} = 0.022 < 0.1$

(iii) Determine the priority vector of Quality Attributes. According to the selection process, we need square the initial matrix first, and then calculate the priority vector.

1	2	5	3
0.5	1	3	2
0.2	0.333	1	0.333
0.333	0.5	3	1

Figure 5.2 Initial weighted matrix

4	7.167	25	11.667
2.267	4	14.5	6.5
0.678	1.233	4	1.933
1.517	2.667	9.167	4

Figure 5.3 Squared weighted matrix

$$\begin{cases} \frac{4}{47.834}, \frac{7.167}{47.834}, \frac{25}{47.834}, \frac{11.667}{47.834} \\ \frac{2.267}{27.267}, \frac{4}{27.267}, \frac{14.5}{27.267}, \frac{6.5}{27.267} \\ \frac{0.678}{7.844}, \frac{1.233}{7.844}, \frac{4}{7.844}, \frac{1.933}{7.844} \\ \frac{1.517}{17.351}, \frac{2.667}{17.351}, \frac{9.167}{17.351}, \frac{4}{17.351} \end{cases}$$

$$= =$$

$$\begin{cases} 0.084, & 0.15, & 0.523, & 0.244 \\ 0.083, & 0.147, & 0.532, & 0.238 \\ 0.086, & 0.157, & 0.51, & 0.246 \\ 0.087, & 0.154, & 0.528, & 0.231 \end{cases}$$

Based on the above schema, the approximate priority vector $\vec{p}$ = (0.085, 0.152, 0.523, 0.24).

(iv) Calculate the total scores for each architecture styles and suggest the appropriate style

QAs / Arch Styles	Usability	Maintainability	Cost	Scalability
SOA (A)	0	1	1	1
MVC (B)	1	1	0	-2

Figure 5.4 The weighted value of each Quality Attribute based on Table 4.2

The result of total score for each style are calculated by using the following formula

$$\omega_{total} = \omega_1 + \omega_2 + \omega_3 + \omega_4$$

The total score for SOA style is:

$$\omega_{total}(A) = (0 * 0.085) + (1 * 0.152) + (1 * 0.523) + (1 * 0.24) = 0.915$$

The total score for MVC style is:

$$\omega_{total}(B) = (1 * 0.085) + (1 * 0.152) + (0 * 0.523) + (-2 * 0.24) = -0.243$$

From the total scores of two architecture styles, it is clear that SOA gets a higher score than the other, thus, SOA is the appropriate style for the b2b application.

6. DISCUSSION

In the Result section, we have presented a complete selection process, which is supported by sufficient literature data and effective mathematical model.

After reviewing several literature (Moaven et al., 2008a; Babu et al., 2010; Vijayalakshmi et al., 2010; Galster et al., 2010) about architecture styles evaluation and selection, we adopt Quality Attributes as the criterion for measuring software architecture styles. We gather 16 common Quality Attributes with brief descriptions in Table 4.1 and it covers most quality requirements of a software system. All these Quality Attributes would be indicators for measuring each architecture style, and we recommend our audiences read it before prioritising Quality Attributes for their software system.

A precondition of selecting proper architecture style is collecting a number of architecture styles that are commonly used today. We collect 14 common architecture styles and categorize them in 6 groups based on their applicable domain. And then each architecture style are measured with every Quality Attribute listed in Table 4.1, we can see any style's performance with any single Quality Attribute. In order to represent information in a more readable way, we integrated data of architecture styles collection and architecture style evaluation, and then we put them in Table 4.2. We can see that both benefits and limitations are being quantified for later comparison. All the measurement is based on literature study on each architecture style.

Software architects and key stakeholders are the people who consider what Quality Attributes should the system achieve (Bass et al., 2003, p.15). It is unlikely that all 16 Quality Attributes are considered, so they should prioritise a number of attributes. Thus, when measuring whether a style is proper to the system or not, the marks of unconsidered Quality Attributes should not affect the selection when marks of considered Quality Attributes do. Moreover, we can have a mathematical model to help with decision making since there are quantified marks in Table 4.2. The AHP model is a proper model that can address the problem. Hence we design a selection process where all mentioned factors are in consideration. With this selection method, we can easily select an architecture style that is probably the best appropriate style for the target system.

6.1 Positioning contribution

The selection of architecture styles is usually based on the expertise and experience of software architects (Qian et al. 2008, p.270). We believe that there is a criteria as well as a process within architects' mind when they deal with architecture style selection. Our major contribution is that we provide a visible criterion and selection process with ease of use to help people who lack expertise and experiences to select an appropriate architecture style systematically.

Some researches has investigated this area, but by comparison with some similar researches, our research have three advantages:

First, we collected more number of common architecture styles. It means that we offer more options to the

audiences. On the contrary, several researches (Moaven et al., 2008a; Babu et al., 2010; Vijayalakshmi et al., 2010; Galster et al., 2010) did not provide over five typical candidate styles for their audience. Clearly our research focuses more on the availability of the selection.

Second, a special point of our research is that we categorize collected architecture styles based on their applicable domain. With these categories, when the audiences understand the nature of the target system, they already got several candidate styles that possibly fit their system's requirements. It means that we avoid unnecessary comparison and computation. We have seen that in some articles (Galster et al., 2010; Moaven et al., 2008a), researchers put 5 architecture styles together without category, which contains Pipe&Filter architecture and Layered architecture (quite different styles), and then they calculate scores for all 5 styles and the one with the highest score turns out to be the most appropriate style, their selection has faults sometimes. In our research, this issue is naturally addressed because of our categorization.

Last but not least, we provide more clear process to apply the AHP model. Each step of our selection process with AHP model is explained quite clearly in our paper. An audience with certain mathematical knowledge can easily utilize our method without checking other articles about the AHP model. We simplified formulas and represent steps in a readable way, so that we have less complexity than others.

6.2 Limitation of our study

The major limitations of our study are that we depend on literature too much so that the reliability is affected in the following ways.

One of the limitations is that the categorization was based on literature study and lack of practical experience. Despite that our categorization avoid some issues that can be met by other researches, however if the categorization is not accurate enough, it will lead to that improper styles become candidates, and then the selection result has lower reliability.

In order to achieve high reliability, we eliminate some Quality Attributes and Architecture styles before displaying them because of a lack of literature support. This elimination reduces the range of our research.

Moreover, because the limitation of time, we do not conduct validation with experts to correct our data. For example, in Table 4.2, all marks are depends on our literature study, it lacks some realistic with practical experience.

7. CONCLUSION & FUTURE WORK

Software Architecture style has been mentioned more and more in software development today. Architecture style selection is the crucial phase in software design because satisfying Quality Attributes is one important issue in software system design that suitable software architecture can fulfill it (Moaven et al., 2008a). This paper exposes a key element of architecture design: Quality Attributes, and uses Quality Attributes as the criterion to measure a number of commonly used architecture styles in categories; with a systematic selection process powered by Analytic Hierarchy Process (AHP) and ends by finding out an appropriate style for target system. It is an effective method with ease of use for people who lack expertise and experience to get proper architecture style for their software system.

Although this paper has some limitations, it paves a way to continue our work with the same research questions. In the future, we would like to extend the number of both Quality Attributes and Architecture Styles, so that this research can cover larger range and be available for more researchers and types of system. We also want contact experts within this domain, e.g. software architects, and conducts interviews with them in order to validate and correct our data, especially the way of categorization and marks in Table 4.2. In addition, we want to include more criteria besides Quality Attributes in order to improve the reliability of selection, since architecture styles is not only decided with Quality Attributes.

ACKNOWLEDGEMENT

The authors of this paper would like to thank all teachers of SE&M for three years support and help. Special thanks to Lennart Petersson for his time and help to this thesis work.

REFERENCES

Babu, D., K., Rajulu, G., P., Reddy, R., A., Kumari, A., A., N., 2010. Selection of Architecture Styles using Analytic Network Process for the Optimization of Software Architecture. *International Journal of Computer Science and Information Security*, Vol. 8, No. 1, April 2010.

Bass, L., Clements, P., Kazmen, R., 2003. *Software Architecture in Practice.* 2nd ed. Boston: Addison-Wesley. Ch.1.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M., 2001. Pattern-Oriented Software Architecture, A system of Patterns. Vol. 8, Siemens AG, Germany.

Coyle, G., 2004. The Analytic Hierarchy Process (AHP). *Practical Strategy, Open Access Material, AHP.*
Available at <http://www.booksites.net/download/coyle/student_files/AHP_Technique.pdf> [Assessed 6 May 2012].

Galster, M., Eberlein, A., Moussavi, M., 2010. Systematic selection of software architecture styles. *IET Softw*., 2010, Vol. 4, lss. 5, pp. 349-360.

Ishizaka, A., Lusti, M., 2006. How to derive priorities in AHP: a comparative study.

Microsoft, 2009. *Microsoft Application Architecture Guide.* 2nd ed. Microsoft Press. Ch.1; Ch.3; Ch.16.

Moaven, S., Habibi, J., Ahmadi, H., Kamandi, A., 2008a. A Decision Support System for Software Architecture-Style Selection. *Sixth International Conference on Software Engineering Research, Management and Applications*.

Moaven, S., Habibi, J., Ahmadi, H., Kamandi, A., 2008b. A Fuzzy Model for Solving Architecture Styles Selection Multi-Criteria Problem. *Second UKSIM European Symposium on Computer Modeling and Simulation.*

Northrop, L., 2003. The Importance of Software Architecture. Software Engineering Institute, Carnegie Mellon University.
Available at <http://csse.usc.edu/gsaw/gsaw2003/s13/northrop.pdf> [Accessed 5 May 2012].

Qian, K., Fu, X., Tao, L., Xu, C., Diaz-Herrera, J., 2008. *Software Architecture and Design Illuminated*. Sudbury, Mass.: Jones and Bartlett Publishers. Ch.1; Ch.5; Ch.6; Ch.7; Ch.9; Ch.10; Ch.12.

Qin, Z., Xing, J., Zheng, X., 2007. *Software Architecture*. Zhejiang University Press. Ch.1; Ch.2; Ch.3.

Saaty, T.L., 1980. The Analytical Hierarchy Process, McGraw-Hill, New York, NY.

Satty, T. L., 2004. Decision making - The analytic hierarchy and network processes (AHP/ANP). *Journal of systems science and systems engineering*, Vol. 13, No. 1, pp. 1-35.

Satty, T. L., 2005. Making and validating complex decisions with the AHP/ANP. *Journal of systems science and systems engineering*, Vol. 14, No. 1, pp. 1-36.

Satty, T. L., 2008. Decision making with the analytic hierarchy process. *Int. J. Services Science*, Vol. 1, No. 1, 2008.

Svahnberg, M., Wohlin, C., Lundberg, L., Mattsson, M., 2002. A Method for Understanding Quality Attributes in Software Architecture Structures. SEKE '02, July 15-19, 2002, Ischia, Italy.

Vijayalakshmi, S., Zayaraz. G., Vijayalakshmi. V., 2010. Multicriteria Decision Analysis Method for Evaluation of Software Architecture. *2010 International Journal of Computer Applications* (0975 - 8887) Vol.1, No. 25.

Zhu, H., 2005. *Software Design Methodology: from principles to architectural styles*. Butterworth-Heinemann. Ch.7.