Fuzzy based Quantitative Evaluation of Architectures using Architectural Knowledge

C. Dhaya and G. Zayaraz

Pondicherry Engineering College, Puducherry, India – 605 014
dhaya.c@gmail.com, gzayaraz@pec.edu

Abstract

Software architecture is a standard and essential part of system development. The knowledge produced and consumed during this process needs to be shared and reused among different stakeholders. Therefore, software architecture knowledge (AK) needs to be managed for improving the architectural capabilities. ADUAK (Architectural Development using Architectural Knowledge) model has been developed to support a framework for capturing and using architectural knowledge to improve the architecture design process. ADUAK guides the architect in selecting the most appropriate architecture, where the interactions between quality attributes, tactics should be analyzed and quantified and the results should be considered for the quality driven architectural design process. Ontology based AK tool is needed, because ontologies are used to combine the real world domain information and software architectural knowledge from various experts, to maintain up – to – date documentation that evolves over time. The primary focus of ADUAK is to acquire knowledge for the fuzzy based evaluation of the architectures of the system using ontologies. It is needed because the architecture represents an enormous risk in a development project. Making bad architectural designs may substantially slow down the system and even lead to failure. The quantitative impact of incorporating the quality in the software architectures is then calculated through a fuzzy based approach for the set of design alternatives. The paper aims to bring the contribution of ADUAK model for the fuzzy based quantitative evaluation of architectures with the help of architectural knowledge through a suitable case study.

Keywords: *Architectural knowledge, Ontology, Software Architectures, Fuzzy logic, Quality attributes*

1. Introduction

Now-a-days, the software architecture [1] research community puts more importance on ‘architectural knowledge’. The so called architectural knowledge concerns the set of design decisions and their rationale [2]. The design decisions are the inherent part of the software architecture [3]. This architectural knowledge plays a major role during architectural development, where architects, developer, and other stakeholders must communicate about the system to be developed.

As the size and complexity of systems increases, more people get involved. There often is not a single all-knowing architect. Instead, this role is fulfilled by more than one person. To be able to take well-founded decisions, all stakeholders need to have the relevant AK at the right place, at the right time. The architectural knowledge is not just represented in the form of decisions and rules, but also includes metaphors, attitudes, rationale to produce successful outcomes. AK needs techniques and tool support to effectively manage the knowledge [4]. ADUAK [5] provides a repository-based model support for theoretic support and guidance in the development process. The repository contains information on the ‘best practices’,

technology preferences, standards, and expertise advise to architectural directions that match with the development processes. The repository is generally an intelligent knowledge base, which is being developed through practice and experiential learning.

Akermann and Tyree worked on the design decisions and exploited them to build an ontology, to support the use of ontology in the software design phase [6]. This ontology based representation yields competitive advantage enabling greater knowledge sharing and reuse. Ontologies can be used to support requirements, traceability between design decisions, specifying quality requirements, predicting quality fulfillment from architectural designs, and finally, measuring quality aspects, which help the software engineers understand the relations and dependencies among various software artifacts.

Architects take for granted the acquaintance with textbooks or former decisions and designs. What is at stake here is a different kind of knowledge, architectural knowledge (AK). What architects really seek are some clues, some hints or well-grounded practical guidelines that refer to the problem they have before them when they put the question or start the query. In this regard, the design of ontologies requires not only representing the written documents (decisions, quality attributes, scenarios, etc.) but also those chunks of architectural knowledge from the daily practice during software development.

The focus of the work is the retrieval of architectural knowledge that helps the architect in the decision based architectural design process [7]. To achieve this goal, we look for related knowledge for the set conceivable notional design alternatives, and the factors which lead to the quality of the designs. The fuzzy based quantitative method for evaluating software architectures is used to select the preferred alternative and help the novice participants to analyze, retrieve, learn and reuse software assets.

The paper is organized as follows. Section 2 describes the formal structure of the ADUAK meta-model and the factors that has maximal articulation in the architectural knowledge. Section 3 describes how the decision making process is applied for different design alternatives with the prototype model developed followed by the demonstration through a case study in Section 4. Section 5 specifies how a quantitative evaluation model is applied for selecting the preferred conceptual design from a set of alternatives under various multi criteria's using Fuzzy AHP. Section 6 shows the tabulated results for selecting the architectures. Section 7 concludes the work of the paper.

2. Meta -Model of ADUAK

Based on theoretical base ideas we propose the following Meta model of ADUAK. This meta-model for software architectures' represents the relations between the decisions, stakeholders involved and the quality requirement. The model models the concepts of quantitative analysis of architectures is represented above in Figure 1. For the architecting process, the main concepts are the requirements, stakeholders, architectural decisions, and the quality factors [8]. Taking into consideration of the stakeholders' functional/quality requirements, the ADUAK makes the decisions which are reflected in the selection of the architectures.

The dependencies between the decisions [9] and the type of decisions related to software architecture, such as the architectural styles and the patterns [10] are included in the model. Quality factors suitable for selecting the design alternatives [11] are also considered. It defines the solutions considered and the one selected upon for the given problem definition. The selection is based upon a quantitative fuzzy model which satisfies the given nonfunctional requirements. The goal of quantitative analysis is to investigate the quality of

different design options, or Architectural solutions. The analysis function is related to the architectures' behavior that can be related to the quality attributes.

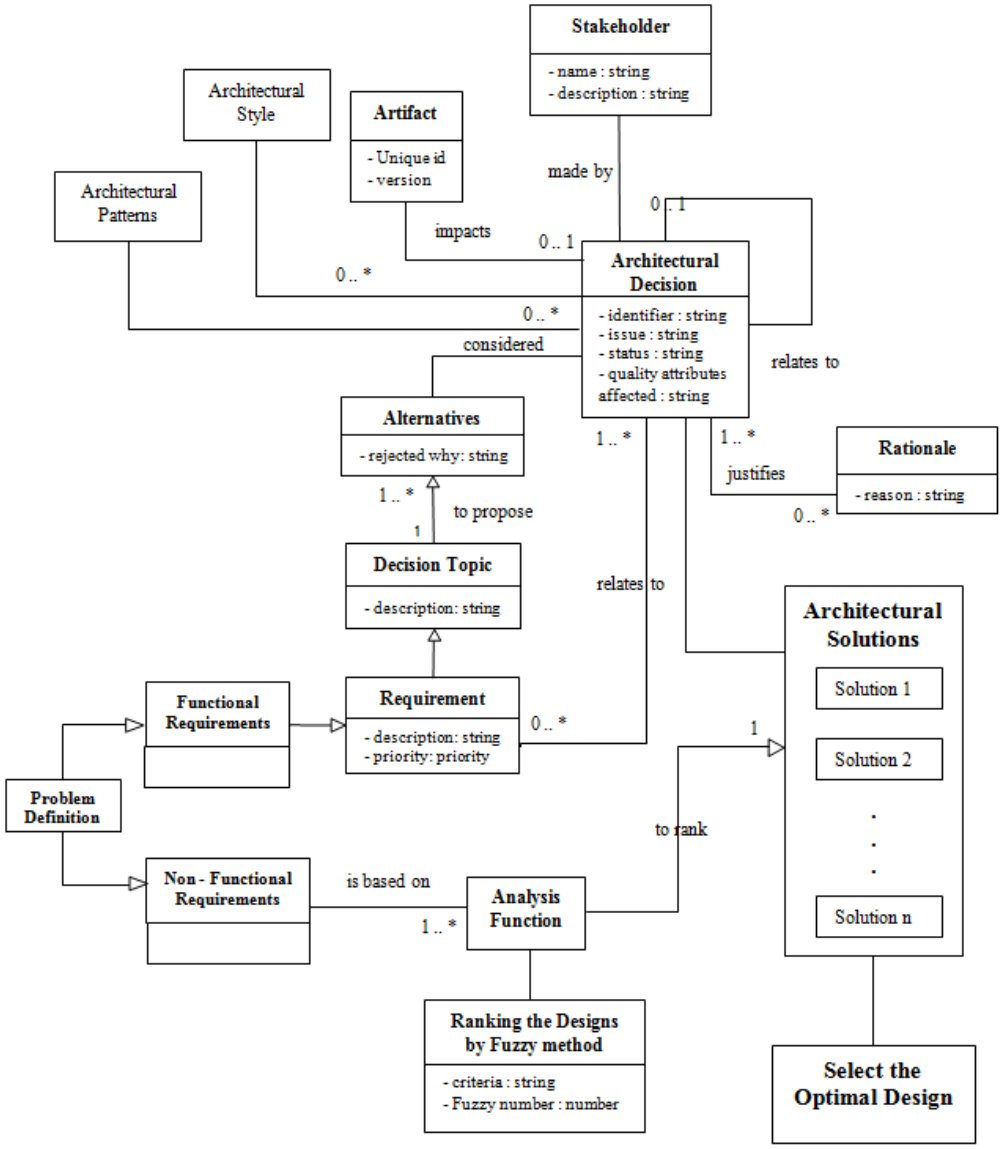

Figure 1. Meta Model of ADUAK

3. Decision Making with Architectural Knowledge

The main motivation of this framework is to provide a systematic approach of using Architecture Knowledge for improving the organization's architectural activities and reusing software assets. The ADUAK model entails the use of both implicit and explicit knowledge [12] in the development of architectures. In order to provide an interactive model for evaluating architectures using architectural knowledge, ADUAK model is developed.As represented in Figure 2, it incorporates the knowledge from previously solved similar examples and the use of expertise advices, experiences in the architecture knowledge repository. The knowledge repository is logically divided into two types of knowledge [13]:

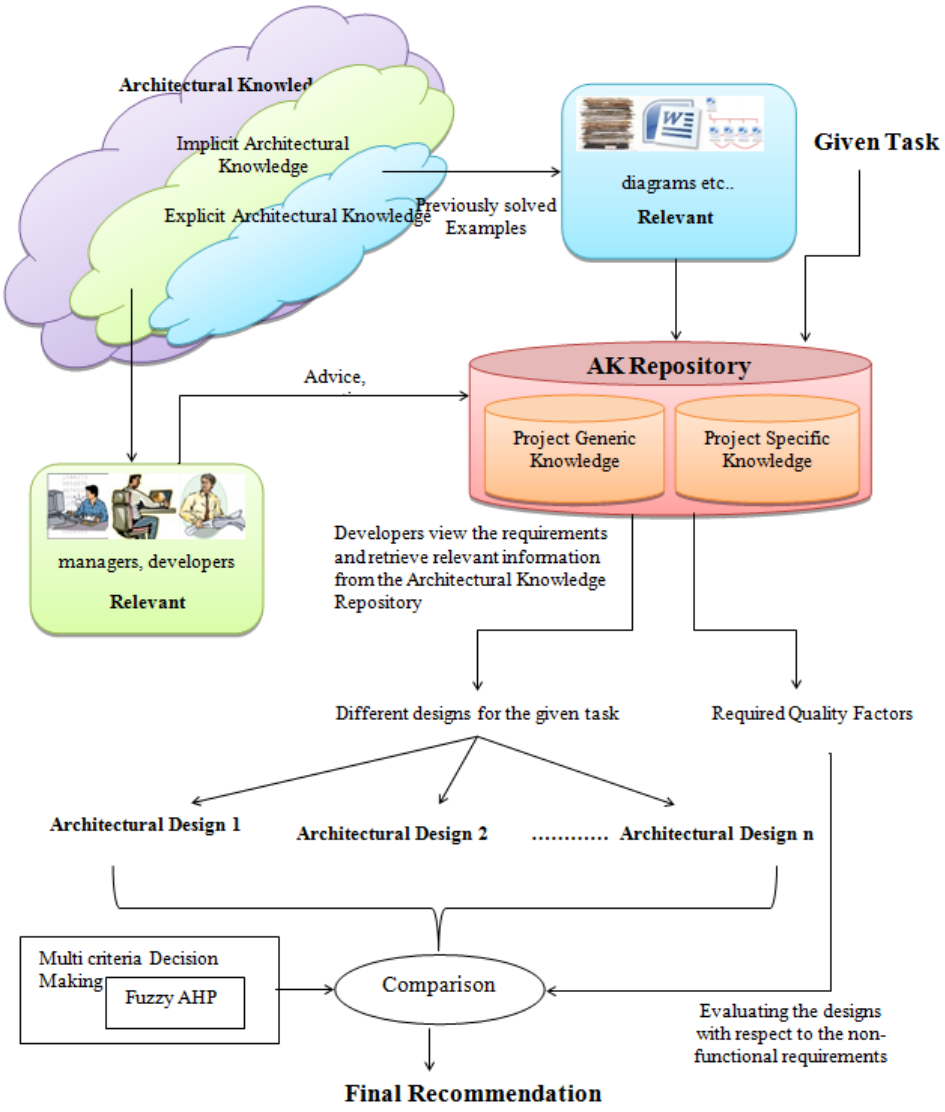

Figure 2. ADUAK Model for Choosing Designs using Architectural Knowledge

Generic architectural knowledge is collected with the general knowledge capturing techniques. Project-specific architectural knowledge gets the artifacts either from the generic knowledge or created during different activities of the software development process. The knowledge concerning the domain analysis, design options evaluated and design decisions [14] made for the similar domains are useful in the design of architectures. The major constraints with architecture design process are (i) Numerous design decisions with multiple design alternatives (ii) Each design requires evaluation and selection from among the alternatives (iii) Many design alternatives conflict (iv) Multiple stakeholders with competing priorities. It is during architectural design that crucial requirements such as performance, reliability, maintainability, etc., must be addressed. In other words, software architecture is "shaped" by quality requirements. This framework helps the architects to achieve quality requirements during architectural design by evaluating the set of plausible architectural

designs [15]. Techniques for evaluating and assessing architectural designs are completely 'valuable'. The task of achieving these quality attributes to meet competing stakeholders' requirements remains a difficult process. In such a complex process of evaluating design alternatives under certain specified criteria, the fuzzy set theory is a suitable mathematical method for selecting an optimal design.

4. Case Study: Course Registration System

To illustrate the example of ADUAK model, an application of Online Course Registration System is presented in this section. Consider the following scenario: The University would like to develop a new design to replace the older mainframe technology for the course registration system. The new system allows students to register and view the reports at any time. Professors will be able to access the system and teach the registered students. Due to the decrease in the federal funding, the university cannot afford to replace the entire system completely. Since the students' population of the university grows over the time, the volume of student registration is more at the enrollment period. The registration system should not affect the other systems of the university like the HR system, financial system due to this up gradation.

In this scenario, the architect and the developers are seeking advice for other experts and retrieving information from the previously solved similar problems. Based on the knowledge retrieved from the AK repository the architectural design decisions [16] are made. Due to the several reasons, architectural decision-making [17] is a difficult task: (i) requirements are captured in an informal manner (ii) non-functional requirements are difficult to specify (iii) concrete architectural decisions have to be made based upon vague requirements (iv)stakeholders involved have different perspectives.

The introduction of ADUAK proved the need to rely on architectural knowledge for the decision-making process within ontology design. The use of ADUAK speeded up the modeling process, as decisions were more easily reached. The ontology shown in the Figure 3 drive the architecture evaluation process. As the evaluation team gains more experiences, the knowledge in the ontology can be evolved and refined. By this manner, the reusability of AK is achieved with the ontology [18]. The ontology is implemented through the open source tool called protégé [19]. The rules to be considered for the architecture evaluation process are as follows:

Rule 1: The instance of architectural decision making must be taken from the available Architectural_Knowledge class.

Rule 2: The architectural design must be selected from the available Architectural_Style class.

Rule 3: The Quality attribute is selected from the Quality_Characteristics class.

Rule 4: The relationship between architectural decision making and the selection of suitable architectural designs is defined through the consistency rules that have been stated with SWRL – Jess tab [20] of Protégé.

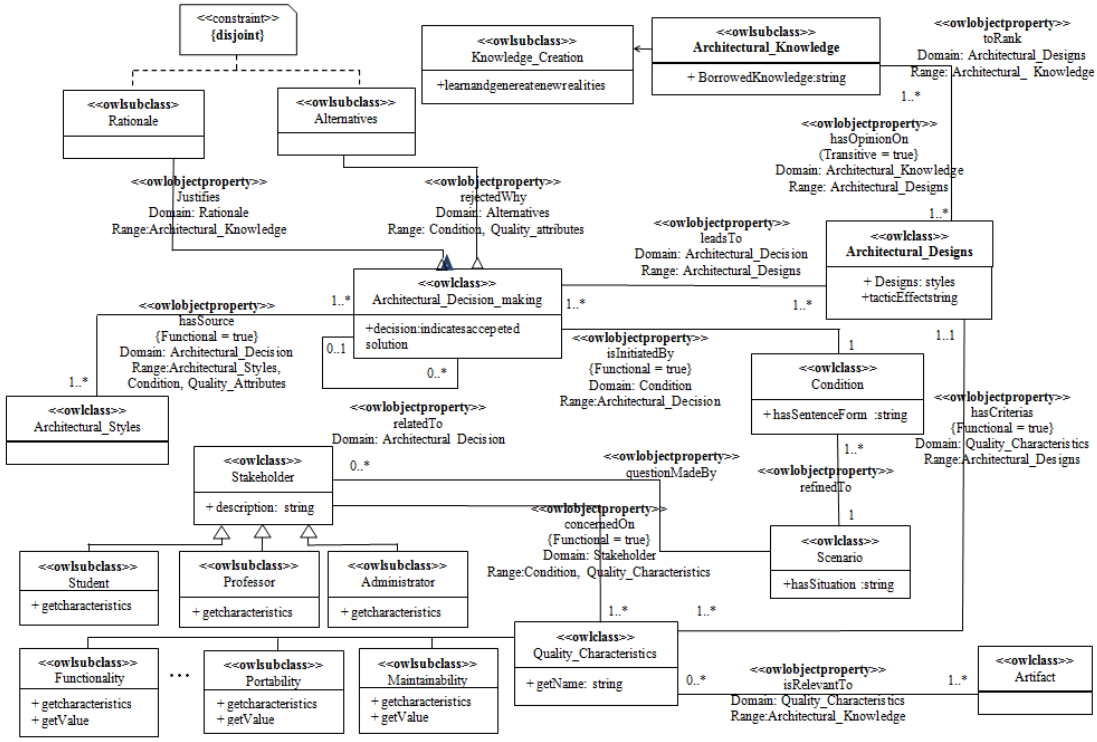

Figure 3. Ontology of the ADUAK Model with Different Object Properties

Consistency Rule 1:
ADUAK :Architectural_Knowledge (?AK) ∧
ADUAK :Architectural_Styles (?AS) ∧
ADUAK :leadBy (?AS, ?AK) ∧
Architectural_Decision_Making (?AD) →Architectural_Style(?AS) ∧ leadBy (?AS, ?AK)
Rule 5: The different Quality_Characteristics have an impact on the Architectural_designs selected through the Architectural_decision_making process.

Consistency Rule 2:
ADUAK :Architectural_Knowledge (?AK) ∧
ADUAK :Architectural_Styles (?AS) ∧
ADUAK : implements(?AS, ?AK) ∧
ADUAK :Quality_Characteristics(?QA) ∧
Xsd:string(?getValueOf)
[Fair, Good, Excellent] (?QA)
ADUAK :getValueOf (?AK, ?QA) ∧
Quality_Characteristics(?QA) ∧ Architectural_Styles (?AS) ∧ Architectural_Decison_Making (?AD) ∧ hasCriterias(?AS, ?AK) →getValueOf(?QA)

The possible 5 architectures like client/server, 3 – tier, thin client, thick client and cloud architectures are suggested by the ADUAK model for the given requirement. ADUAK rank the architecture's thereby maximizing the degree of safety and performance. The ranking is done based on the pros and cons of the each architecture for the given requirement. Each decision will have the evaluated alternatives, tradeoffs and rationale and discussion [21] for

selecting the suitable architecture. If all the relevant architectural knowledge is made explicit, the solution to the problem is identified in an easier manner.

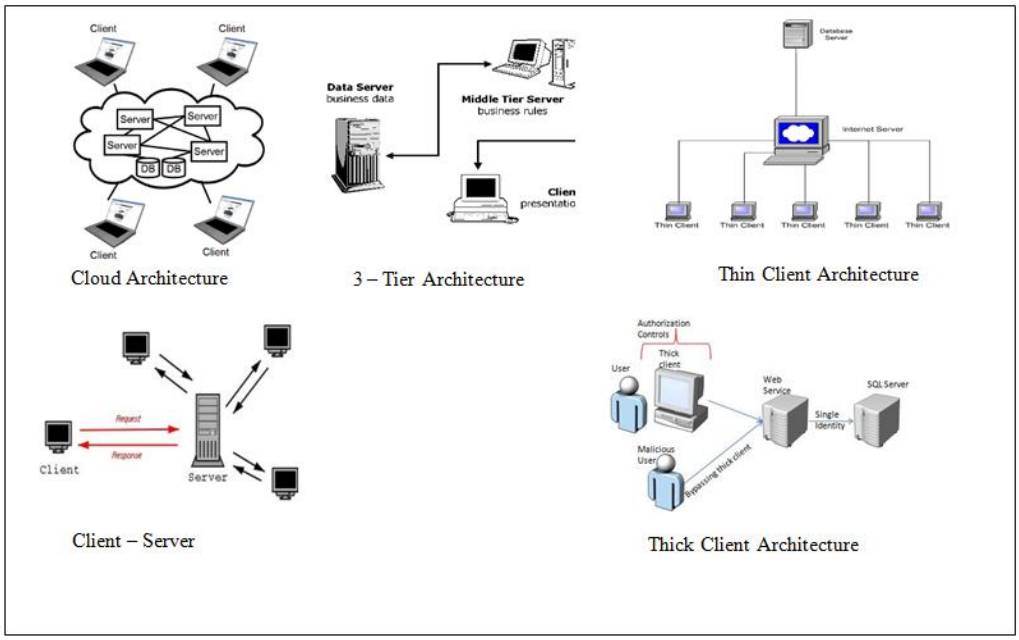

Figure 4. Considered Architectures

The aim of this framework is to give the final recommendation for the architect in the selection of the specific architecture which satisfies the different architectural forces for the given scenario. It is a multiple criteria decision-making framework, which provides a comparison of technical performance, usability and reliability of a system during the evaluation to assist decision-makers in selecting the winning design that best satisfies the requirement in hand. The top ranked competing architectures for the given case study are shown in Figure 4. For the development, the university needs to choose suitable software architecture from several competing candidate architectures.

Choosing a software architecture design remains a complex task. The non-functional requirements should be addressed early so as to select an optimal design. The analysis framework follows ISO 9126 to specify the quality attributes and their sub-characteristics. The model, however, can be adapted [22] to specify any quality attribute. Since different stakeholders have different quality requirements, the extent to which the stakeholders' quality requirement was satisfied is a tedious process. Identifying different users and classifying them is done using segmentation [23]. Segmentation refers to grouping of people based on similar characteristics. Participating persons are given the choice for selecting their relevant quality attributes.

The different users involved and their associated quality attributes for the case study considered are tabulated below in Table 1. If the architecture evaluation method quantitatively analyzes the architectures, it helps the developers in choosing the best option. For the purpose of explanation, the quality requirements of the administrator have been considered. The quality characteristics of the administrator identified using the ranked decisions using ADUAK are displayed in the Table 2.

Table 1. Quality Requirements of Each Group

Group	Quality Attributes considered by different stakeholders
Student	Functionality, Usability
Professor	Functionality, Reliability, Usability
Administrator	Functionality, Reliability, Usability, Efficiency, Maintainability, Portability

Table 2. Quality Attributes for the Considered Architectures

Quality Attributes/ Architectures	Client – Server architecture	Thin Client architecture	Thick Client architecture	3 – tier architecture	Cloud architecture
Functionality	Fair	Good	Good	Good	Excellent
Reliability	Fair	Good	Good	Good	Excellent
Usability	Fair	Good	Good	Fair	Good
Efficiency	Fair	Good	Excellent	Good	Excellent
Maintainability	Fair	Good	Fair	Good	Good
Portability	Good	Good	Good	Good	Excellent

Deciding a suitable architecture among complex criteria's is always associated with uncertainty. The most suitable method of accounting for uncertainty and ambiguity is provided by the fuzzy set theory [24]. In this real world situation, precise data relating to measurement indicators is very difficult to extract from expert judgments. This happens because humans encompass a degree of uncertainty, and decision makers are unable to assign crisp numerical values for comparisons. Decision makers normally prefer natural language expressions over exact numbers when accessing criteria and alternatives. Probabilistic approach can be used when the information regarding the phenomenon is incomplete. Fuzzy set deals with such vagueness, imprecision or not well defined data's. By approximating this uncertain information, where decisions are concerned for alternatives, it effectively resembles human views, ideas and perceptions. Fuzzy theory using the membership functions and the fuzzy numbers effectively transforms vague information into useful data. In the real world, decision-making in the real world where appropriate data and the sequences of possible actions are not quite known, the fuzzy methodology articulates the relative importance of the alternatives and the criteria with fuzzy numbers rather than crisp ones. Fuzzy AHP is a synthetic extension of classical AHP method when the fuzziness of the decision maker is considered.

5. Fuzzy based Quantitative Evaluation of Architectures

Architectural design can be considered as a decision making process influenced by many stakeholders. Considering design as a multi stakeholder decision making process, it reasonable to apply multi criteria decision making (MCDM) techniques [25] to asses design quality. In MCDM it's often need to select or rank alternatives that are associated with intangible or conflicting attributes. In this situation, an analytical way to make a successful decision is needed.

The fuzzy AHP technique can be viewed as an advanced analytical method developed from the traditional AHP. Despite the convenience of AHP in handling both quantitative and qualitative criteria of multi-criteria decision making problems, Laarhoven have studied the fuzzy AHP [26] which is the extension of Saaty's theory. Fuzzy AHP shows relatively more sufficient description in decision making processes compared to the traditional AHP methods.

5.1 Basic Concepts of Fuzzy Set Theory

A fuzzy set Ã in a universe of discourse X is characterized by a membership function $\mu_{\tilde{A}}(x)$, which associates with each element *x* in X a real number in the interval [0, 1]. The function value $\mu_{\tilde{A}}(x)$is termed the grade of membership of *x* in Ã.

$$\mu_{\tilde{A}} : x \rightarrow [0,1] \qquad \text{---------- (1)}$$

Although there are many shapes of fuzzy numbers, the triangular and trapezoidal shapes are used most often for representing fuzzy numbers. The following definitions show that membership function of the triangular fuzzy number, and its operations.

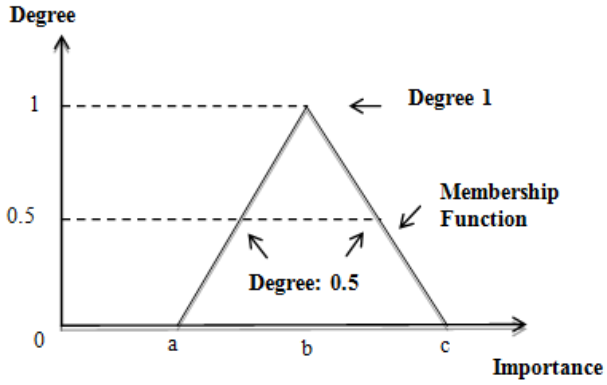

Figure 5. The Sample of Fuzzy Membership Function

Definition 1: A triangular fuzzy number can define as a triplet (a, b, c), as shown in Figure 5. Thus, fuzzy membership function due to the triangular weights is:

$$\mu_{\tilde{A}}(x) = \begin{cases} (x-a)/(b-a) & a \le x \le b \\ (c-x)/(c-b) & b \le x \le c \\ 0 & otherwise \end{cases} \text{---------- (2)}$$

Where b is the most possible value of a fuzzy number Ã, a and c are the lower and upper bounds, respectively before and beyond them the element will have no relationship to the set.

Definition 2: While variables in mathematics usually take numerical values, in fuzzy logic applications, the nonnumeric linguistic variables are often used to facilitate the expression of rules and facts. A linguistic variable is a variable whose values are expressed in linguistic terms. The concept of a linguistic variable is very useful in dealing with situations which are too complex or not well defined to be reasonably described in conventional quantitative expressions [30].

5.2 Steps Involved in Fuzzy AHP

Step 1: Fuzzy Pair-wise Comparison Matrix

The fuzzy Pairwise Comparison Matrix [27] is created using the triangular fuzzy number (a, b, c) is determined using the definition 1 and 2. The conversion is indicated in the Table 3.

Table 3. Triangular Fuzzy PCM

Linguistic Variables	Triangular Fuzzy PCM value	Inverse Triangular Fuzzy PCM value
Equal	(1,1,1) if diagonal; (1,1,3) otherwise	(1,1,1) if diagonal; (1,1,3) otherwise
Very poor	(1,2,4)	(1/4,1/2,1/1)
Poor	(1,3,5)	(1/5,1/3,1/1)
Fair	(2,4,6)	(1/6,1/4,1/2)
Medium Good	(3,5,7)	(1/7,1/5,1/3)
Good	(4,6,8)	(1/4,1/6,1/8)
Very Good	(5,7,9)	(1/9,1/7,1/5)
Medium Excellent	(6,8,10)	(1/10,1/8,1/6)
Excellent	(7,9,11)	(1/11,1/9,1/7)

The triangular fuzzy comparison matrix is defined as follows:
For i,j = 1…n and i ≠ j

$$\tilde{A} = \begin{bmatrix} (1\ 1\ 1) & (a_{12}\ \ b_{12}\ \ c_{12}) & \cdots & (a_{1n}\ \ b_{1n}\ \ c_{n1}) \\ (a_{21}\ \ b_{21}\ \ c_{21}) & (1\ 1\ 1) & & (a_{2n}\ \ b_{2n}\ \ c_{2n}) \\ (a_{n1}\ \ b_{n2}\ \ c_{n2}) & (a_{n2}\ \ b_{n2}\ \ c_{n2}) & \cdots & (1\ 1\ 1) \end{bmatrix} \quad \text{---------- (3)}$$

Step 2: Fuzzy Extent Analysis

Assume X = {*x*1, *x*2, . . . ,*xn*} be the object set, and *G* = { *g*1, *g*2, . . . , *gm*} be the set of goals. According to Chang's extent analysis model, extent analysis for each goal *gi*, is performed for each object *x*i. Therefore, *m* extent analysis values for each object can be obtained with the

$$M_{gi}^{1}, M_{gi}^{2}, \ldots . M_{gi}^{m}, i = 1,2,3\,..n$$

where all $M_{gi}^{j}(j = 1,2,..n)$ are triangular fuzzy numbers.

The value of fuzzy synthetic extent with respect to its object is as follows,

$$s_i = \frac{\sum_{j=1}^{m} M_{gi}^{j}}{\sum_{i=1}^{n}\sum_{j=1}^{m} M_{gi}^{j}} \quad \text{---------- (4)}$$

Fuzzy extent analysis is applied to get the fuzzy performance matrix, which represents the overall performance of all alternatives with respect to each criterion. It can be obtained by multiplying the weighting vector (W) by the decision matrix (X).

$$P = X * W = \begin{bmatrix} w_1 x_{11} & w_2 x_{12} & w_m x_{1m} \\ w_1 x_{21} & w_2 x_{22} & w_m x_{2m} \\ \vdots & \vdots & \vdots \\ w_1 x_{n1} & w_2 x_{n2} & w_m x_{nm} \end{bmatrix} \text{---------- (5)}$$

Step 3: Alpha Cut Analysis Method

An alpha cut (α – cut) [24] is a crisp set of elements of A belonging to the fuzzy set to a degree α. It's needed for checking and comparing fuzzy number. The α-cut based method stated that if let A and B be fuzzy numbers with α-cut, $A_a = [a_\alpha^-, a_\alpha^+]$and $[b_\alpha^-, b_\alpha^+]$. It says that A can be smaller than B if, $a_\alpha^- < b_\alpha^-$ and $a_\alpha^+ < b_\alpha^+$ for all $\alpha \epsilon [0,1]$.

The α – cut analysis transforms the total weighted performance matrices into interval performance matrices, as represented in α_{Left} and α_{Right} for each alternatives as follows.

$$\tilde{x}_\alpha = \begin{bmatrix} \alpha_{left\,1} & \alpha_{right1} \\ \alpha_{left2} & \alpha_{right2} \\ \vdots & \vdots \\ \alpha_{leftn} & \alpha_{rightn} \end{bmatrix} \text{---------- (6)}$$

$$\alpha_{left} = [\alpha * (b - a)] + a \text{---------- (7)}$$

$$\alpha_{right} = c - [\alpha * (c - b)] \text{---------- (8)}$$

a, b, c are lower, middle and upper fuzzy respectively.

Step 4: λ Function and Crisp Value Normalization

α_{left} and α_{right} need to be converted to a crisp value which is done by λ function. λ function is used to represent the choice of the decision maker ie. Optimistic, pessimistic or moderate. Decision maker with optimistic attribute chooses the medium λ and the pessimistic person chooses the minimum λ.

$$Crisp\,Value = \lambda * \alpha_{right} + [(1 - \lambda) * \alpha_{left})] \text{---------- (9)}$$

$$Crisp\,Value\,(C_\lambda) = \begin{bmatrix} C_{1\lambda} \\ C_{2\lambda} \\ \vdots \\ C_{i\lambda} \end{bmatrix} \text{---------- (10)}$$

where λ = [0,1]

Finally, the crisp value needs to be normalized, because the elements are in different values.

$$C_{i\lambda} = \frac{C_{i\lambda}}{\sum C_{i\lambda}} \text{---------- (11)}$$

6. Evaluation of Architectures for the Given Case Study

Step 1: The fuzzy PCM is calculated for different criteria's and architectural designs using the Table 3.

The Fuzzy matrix is represented in Table 4 to Table 10.

Table 4. Fuzzy Pairwise Comparison for Different Quality Attributes

Quality Attributes	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability
Functionality	(1,1,1)	(1,2,4)	(1,3,5)	(2,4,6)	(3,5,7)	(5,7,9)
Reliability	(1/4,1/2,1/1)	(1,1,1)	(1,3,5)	(1,3,5)	(3,5,7)	(5,7,9)
Usability	(1/5,1/3,1/1)	(1/5,1/3,1/1)	(1,1,1)	(1/4,1/2,1/1)	(1,3,5)	(3,5,7)
Efficiency	(1/6,1/4,1/2)	(1/5,1/3,1/1)	(1,2,4)	(1,1,1)	(1,3,5)	(3,5,7)
Maintainability	(1/7,1/5,1/3)	(1/7,1/5,1/3)	(1/5,1/3,1/1)	(1/5,1/3,1/1)	(1,1,1)	(1,3,5)
Portability	(1/9,1/7,1/5)	(1/9,1/7,1/5)	(1/7,1/5,1/3)	(1/7,1/5,1/3)	(1/5,1/3,1/1)	(1,1,1)

Table 5. Fuzzy PCM of Architectural Designs for the Quality Attribute 1: Functionality

Architectures	Client – Server architecture	Thick client architecture	Thin Client architecture	3 – tier architecture	Cloud architecture
Client – Server architecture	(1,1,1)	(1/7,1/5,1/3)	(1/7,1/5,1/3)	(1/7,1/5,1/3)	(1/9,1/7,1/5)
Thick Client architecture	(3,5,7)	(1,1,1)	(1,3,5)	(1,3,5)	(1/9,1/7,1/5)
Thin Client architecture	(3,5,7)	(1/5,1/3,1/1)	(1,1,1)	(1,3,5)	(1/7,1/5,1/3)
3 – tier architecture	(3,5,7)	(1/5,1/3,1/1)	(1/5,1/3,1/1)	(1,1,1)	(1/9,1/7,1/5)
Cloud architecture	(5,7,9)	(5,7,9)	(3,5,7)	(5,7,9)	(1,1,1)

Table 6. Fuzzy PCM of Architectural Designs for the Quality Attribute 2: Reliability

Architectures	Client – Server architecture	Thick client architecture	Thin Client architecture	3 – tier architecture	Cloud architecture
Client – Server architecture	(1,1,1)	(1/7,1/5,1/3)	(1/7,1/5,1/3)	(1/7,1/5,1/3)	(1/9,1/7,1/5)
Thick Client architecture	(3,5,7)	(1,1,1)	(1,3,5)	(1,3,5)	(1/9,1/7,1/5)
Thin Client architecture	(3,5,7)	(1/5,1/3,1/1)	(1,1,1)	(1,3,5)	(1/7,1/5,1/3)
3 – tier architecture	(3,5,7)	(1/5,1/3,1/1)	(1/5,1/3,1/1)	(1,1,1)	(1/9,1/7,1/5)
Cloud architecture	(5,7,9)	(5,7,9)	(3,5,7)	(5,7,9)	(1,1,1)

Table 7. Fuzzy PCM of Architectural Designs for the Quality Attribute 3: Usability

Architectures	Client – Server architecture	Thin client architecture	Thick Client architecture	3 – tier architecture	Cloud architecture
Client – Server architecture	(1,1,1)	(1/6,1/4,1/2)	(1/6,1/4,1/2)	(1/4,1/2,1/1)	(1/7,1/5,1/3)
Thin Client architecture	(2,4,6)	(1,1,1)	(1,3,5)	(3,5,7)	(1,3,5)
Thick Client architecture	(2,4,6)	(1/5,1/3,1/1)	(1,1,1)	(3,5,7)	(1,3,5)
3 – tier architecture	(1,2,4)	(1/7,1/5,1/3)	(1/7,1/5,1/3)	(1,1,1)	(1/7,1/5,1/3)
Cloud architecture	(3,5,7)	(1/5,1/3,1/1)	(1/5,1/3,1/1)	(3,5,7)	(1,1,1)

Table 8. Fuzzy PCM of Architectural Designs for the Quality Attribute 4: Efficiency

Architectures	Client – Server architecture	Thin client architecture	Thick Client architecture	3 – tier architecture	Cloud architecture
Client – Server architecture	(1,1,1)	(1/4,1/2,1/1)	(1/5,1/3,1/1)	(1/4,1/2,1/1)	(1/5,1/3,1/1)
Thin Client architecture	(1,2,4)	(1,1,1)	(1,3,5)	(1,2,4)	(1,3,5)
Thick Client architecture	(1,3,5)	(1/5,1/3,1/1)	(1,1,1)	(1,3,5)	(2,4,6)
3 – tier architecture	(1,2,4)	(1/4,1/2,1/1)	(1/5,1/3,1/1)	(1,1,1)	(1/4,1/2,1/1)
Cloud architecture	(1,3,5)	(1/5,1/3,1/1)	(1/6,1/4,1/2)	(1,2,4)	(1,1,1)

Table 9. Fuzzy PCM of Architectural Designs for the Quality Attribute 5: Maintainability

Architectures	Client – Server architecture	Thin client architecture	Thick Client architecture	3 – tier architecture	Cloud architecture
Client – Server architecture	(1,1,1)	(1/6,1/4,1/2)	(1/5,1/3,1/1)	(1/6,1/4,1/2)	(1/7,1/5,1/3)
Thin Client architecture	(2,4,6)	(1,1,1)	(1/5,1/3,1/1)	(1/4,1/2,1/1)	(1/5,1/3,1/1)
Thick Client architecture	(1,3,5)	(1,3,5)	(1,1,1)	(1,3,5)	(1,2,4)
3 – tier architecture	(2,4,6)	(1,2,4)	(1/5,1/3,1/1)	(1,1,1)	(1/5,1/3,1/1)
Cloud architecture	(3,5,7)	(1,3,5)	(1/4,1/2,1/1)	(1,3,5)	(1,1,1)

Table 10. Fuzzy PCM of Architectural Designs for the Quality Attribute 6: Portability

Architectures	Client – Server architecture	Thin client architecture	Thick Client architecture	3 – tier architecture	Cloud architecture
Client – Server architecture	(1,1,1)	(1/5,1/3,1/1)	(1/5,1/3,1/1)	(1/5,1/3,1/1)	(1/6,1/4,/12)
Thin Client architecture	(1,3,5)	(1,1,1)	(1/4,1/2,1/1)	(1,3,5)	(1/5,1/3,1/1)
Thick Client architecture	(1,3,5)	(1,2,4)	(1,1,1)	(1/4,1/2,1/1)	(1/5,1/3,1/1)
3 – tier architecture	(1,3,5)	(1/5,1/3,1/1)	(1,2,4)	(1,1,1)	(1/5,1/3,1/1)
Cloud architecture	(2,4,6)	(1,3,5)	(1,3,5)	(1,3,5)	(1,1,1)

Step 2: After obtaining the fuzzified pairwise comparison matrix, the fuzzy extent analysis is applied using the formula (4).

For the fuzzified pairwise comparison of criteria's, using the Table 4, the total fuzzy PCM is calculated as,

Left = 40.65(b_1)
Middle = 70.31 (b_2)
Right = 106.22 (b_3)

First row sum for the Quality Attribute 1: Functionality is calculated using the Table 5

Left = 13 (a_1)
Middle = 22 (a_2)
Right = 32 (a_3)

First row sum / Total sum

Left = a_1 / b_3 = 0.1224
Middle = a_2 / b_2 = 0.3129
Right = a_3 / b_1 = 0.7872

The same calculation is applied for other criteria's: Reliability, Efficiency, Usability, Maintainability & Portability, which is calculated using the Table 6, Table 7, Table 8, Table 9 and Table 10 respectively. The fuzzy weighted performances of different quality attributes are displayed in Table 11.

Table 11. Fuzzy Extent Analysis of Different Quality Attributes

	Total Sum			First row sum			$Fuzzy\ Performances = \frac{First\ row\ sum}{Total\ sum}$		
	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right
Functionality	40.65	70.31	106.22	13	22	32	0.1224	0.3129	0.7872
Reliability				11.25	19.5	28	0.1059	0.2773	0.6888
Usability				5.65	10.16	16	0.0532	0.1445	0.3936
Efficiency				6.37	11.58	18.5	0.0600	0.1647	0.4551
Maintainability				2.68	5.06	8.66	0.0252	0.0720	0.2130
Portability				1.7	2.01	3.06	0.0160	0.0286	0.0753

Similarly fuzzy extent analysis for competing architectures is calculated and is tabulated below in Table 12. The Weighted performance of the competing architectures using the Quality attributes is displayed in Table 13 and Table 14.

Table 12. Fuzzy Extent Analysis of Competing Architectures

	Functionality			Reliability			Usability			Efficiency			Maintainability			Portability		
	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right
Client Server architecture	0.0200	0.0300	0.0600	0.0200	0.0300	0.0600	0.0200	0.0500	0.1200	0.0300	0.0700	0.2800	0.0300	0.0500	0.1600	0.0300	0.0600	0.2500
Thin Client architecture	0.0800	0.2100	0.5000	0.0800	0.2100	0.5000	0.1100	0.3400	0.9000	0.0800	0.3100	1.0500	0.0600	0.1500	0.4800	0.0500	0.2100	0.7200
Thick Client architecture	0.0700	0.1700	0.3900	0.0700	0.1700	0.3900	0.1000	0.2800	0.7500	0.0800	0.3200	0.9900	0.0800	0.3000	0.9500	0.0500	0.1800	0.6600
3 – tier architecture	0.0600	0.1200	0.2800	0.0600	0.1200	0.2800	0.0300	0.0800	0.2200	0.0400	0.1200	0.4400	0.0700	0.1900	0.6200	0.0500	0.1800	0.6600
Cloud architecture	0.2400	0.4700	0.9600	0.2400	0.4700	0.9600	0.1100	0.2500	0.6400	0.0500	0.1800	0.6300	0.1000	0.3100	0.9100	0.0900	0.3700	1.2200

Table 13. Weighted Performance of Competing Architectures using the Quality Attributes (WithFormula 5 and using Table 11 & 12)

	Functionality			Reliability			Usability			Efficiency			Maintainability			Portability		
	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right	Left	Middle	Right
Client – Server architecture	0.0024	0.0094	0.0472	0.0021	0.0083	0.0413	0.0011	0.0072	0.0472	0.0018	0.0115	0.1274	0.0008	0.0036	0.0341	0.0005	0.0017	0.0188
Thin Client architecture	0.0098	0.0657	0.3936	0.0085	0.0582	0.3444	0.0059	0.0491	0.3542	0.0048	0.0511	0.4779	0.0015	0.0108	0.1022	0.0008	0.0060	0.0542
Thick Client architecture	0.0086	0.0532	0.3070	0.0074	0.0471	0.2686	0.0053	0.0405	0.2952	0.0048	0.0527	0.4505	0.0020	0.0216	0.2024	0.0008	0.0051	0.0497
3 – tier architecture	0.0073	0.0375	0.2204	0.0064	0.0333	0.1929	0.0016	0.0116	0.0866	0.0024	0.0198	0.2002	0.0018	0.0137	0.1321	0.0008	0.0051	0.0497
Cloud architecture	0.0294	0.1471	0.7557	0.0254	0.1303	0.6612	0.0059	0.0361	0.2519	0.0030	0.0296	0.2867	0.0025	0.0223	0.1938	0.0014	0.0106	0.0919

Table 14. Total Weighted Performance

Architectures	Total Weighted Performance		
	Left	Middle	Right
Client – Server architecture	0.0087	0.0418	0.3161
Thin Client architecture	0.0312	0.2409	1.7266
Thick Client architecture	0.0289	0.2202	1.5734
3 – tier architecture	0.0203	0.1210	0.8819
Cloud architecture	0.0676	0.3761	2.2413

Step 3: Using α – cut analysis, the above weighted performance matrix is converted to performance interval matrix using the formula (7) and (8). The calculated α – cut analysis matrix is displayed in the Table 15.

Table 15. α – cut Analysis Matrix

α - level	Client – server architecture		Thin client architecture		Thick client architecture		3 – tier architecture		Cloud architecture	
	α_{left}	α_{right}	α_{left}	α_{right}	α_{left}	α_{right}	α_{left}	α_{right}	α_{left}	α_{right}
0.1	0.0120	0.2887	0.0522	1.5780	0.0480	1.4381	0.0303	0.8058	0.0985	2.0548
0.2	0.0153	0.2613	0.0732	1.4294	0.0672	1.3028	0.0404	0.7297	0.1293	1.8682
0.3	0.0186	0.2338	0.0941	1.2809	0.0863	1.1675	0.0505	0.6536	0.1601	1.6817
0.4	0.0219	0.2064	0.1151	1.1323	0.1054	1.0322	0.0605	0.5775	0.1910	1.4952
0.5	0.0252	0.1790	0.1361	0.9837	0.1246	0.8968	0.0706	0.5014	0.2218	1.3087
0.6	0.0285	0.1515	0.1571	0.8352	0.1437	0.7615	0.0807	0.4253	0.2527	1.1221
0.7	0.0318	0.1241	0.1780	0.6866	0.1628	0.6262	0.0908	0.3492	0.2835	0.9356
0.8	0.0352	0.0966	0.1990	0.5381	0.1820	0.4909	0.1008	0.2732	0.3144	0.7491
0.9	0.0385	0.0692	0.2200	0.3895	0.2011	0.3556	0.1109	0.1971	0.3452	0.5626

Step 4: The defuzzification is done through the λ function ie. α_{left} and α_{right} need to be converted to a crisp value. For the above Table 15, the alpha cut is converted into crisp value and it's normalized using the formula (9) and (11). The calculation is tabulated below in Table 16.

Table 16. Fuzzy AHP using α – cut and λ – function

Architectures	α – cut (α = 0.5)		Crisp value		Crisp value (After Normalization)		Rank
	α_{left}	α_{right}	λ = 0.5	λ = 0.7	λ = 0.5	λ = 0.7	
Client – server architecture	0.0252	0.1790	0.1021	0.1329	0.0812	0.0983	**5**
Thin Client architecture	0.1361	0.9837	0.2226	0.2487	0.1771	0.1840	**3**
Thick Client architecture	0.1246	0.8968	0.2965	0.3167	0.2359	0.2342	**2**
3 – tier architecture	0.0706	0.5014	0.1983	0.2070	0.1578	0.1531	**4**
Cloud architecture	0.2218	1.3087	0.4374	0.4468	0.3480	0.3305	**1**

From the table 16, we get the order of the competing alternatives as cloud >> thick client>>thin client >> 3 – tier >> Client – Server architecture. Cloud architecture gains the best score among the alternatives and therefore, it's the optimal solution for the stated objective.

7. Conclusion

Software Architecture is the knowledge intensive process. Architectural Knowledge is defined as the collection of architectural design decisions and architectural forces that influence these decisions. The importance of managing architectural design decisions and its rationale has been gained significant recognition. This paper aims at improving the efficiency of the architectural design process through architectural knowledge management (AKM) support. A meta-model called ADUAK is developed to support the AKM. The success of any system is down to the attention paid to the design process. ADUAK has the potential to help the architects in improving the software architecture process by providing competing design alternatives to design software architecture. An evaluation method will be much more effective if the architectural knowledge of the evaluations is reused by the software architecture team during the architecture development process. Using ontologies we express the vocabularies and semantics of our problem domain with an expressive, explicit and well-defined manner. In this paper, we propose how ADUAK generates the necessary knowledge to drive the evaluation process in an ontological manner. The main objective of ADUAK is to evaluate competing architectures in a quantified manner. A quantitative model for evaluating a set of candidate architectural designs based on a stated objective has been developed and its usage has been illustrated using a simple case study. The developed quantitative model relies on the Fuzzy AHP method for computing relative priorities of quality attributes.

References

[1] L. Bass, P. Clements and R. Kazman, "Software Architecture in Practice", 2nd edition, Addison-Wesley Pearson Education, **(2003)**.
[2] R. Farenhorst, P. Lago and H. van Vliet, "Effective Tool Support for Architectural Knowledge Sharing", Proceedings of 5th European Conference on Software Architecture (ECSA), **(2007)**, pp. 123-138.
[3] A. Jansen and J. Bosch, "Software Architecture as a Set of Architectural Design Decisions", Proceedings of 5thIEEE/IFIP Working Conference on Software Architecture, **(2005)** November, pp. 109-120.
[4] A Tang, P. Avgeriou, A. Jansen, R. Capilla and M. A. Babar, "A comparative study of architecture knowledge management tools", Journal of System and Software, vol. 83, no. 3, **(2010)**, pp. 352-370.
[5] C. Dhaya and G. Zayaraz, " Development of multiple Architectural Designs using ADUAK", Proceedings of the first IEEE International Conference on Communication & Signal Processing, **(2012)** April, pp. 93-97.
[6] A. Akerman and J. Tyree, "Using ontology to support development of software architectures", IBM System Journal, vol. 45, no. 4, **(2006)**, pp. 813-825.
[7] G. Zayaraz, C. Dhaya and V. Vijayalakshmi, "A Survey on Decision based Software Architecture Design Approaches", Journal of Computing, vol. 3, Issue 12, **(2011)** December, pp. 93-101.
[8] R. Farenhorst and R. C. de Boer, "Core Concepts of an Ontology of Architectural Design Decisions", Technical Report IR-IMSE-002, VrijeUniversiteit Amsterdam, **(2006)**.
[9] A. Jansen, J. van der Ven, P. Avgeriou and D. K. Hammer, "Tool Support for Architectural Decisions", Proceedings of the 6th working IEEE/IFIP Conference on Software Architecture, Mumbai, India, **(2007)**.
[10] F. Buschmann, et al., "Pattern-Oriented Software Architecture: A System of Patterns", John Wiley & Sons, **(1996)**.
[11] D. Gross and E. Yu, "From Non-Functional Requirements to Design Through Patterns", Requirements Engineering Journal, vol. 6, **(2001)**, pp. 18-36.
[12] D. Borches and M. Bonnema, "A3 Architecture Overviews - Focusing Architectural Knowledge to Support Evolution of Complex Systems", 20th Annual INCOSE Symposium, **(2010)**.
[13] M. A. Babar, A. Northway, I. Gorton, P. Heuer and T. Nguyen, "Introducing tool support for managing architectural knowledge: an experience report", In: Proceedings of the 15th IEEE International Conference on Engineering Computer-Based Systems (ECBS '08), **(2008)**, pp. 105–113.
[14] M. Ali-Babar, I. Gorton and R. Jeffery, "Toward a Framework for Capturing and Using Architecture Design Knowledge", Tech Report TR-0513, University of New South Wales, Australia, **(2005)**.

[15] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods", IEEE Transactions on Software Engineering, vol. 28, no. 7, **(2002)**, pp. 638–653.
[16] J. S. van der Ven, A. G. J. Jansen, J. A. G. Nijhuis and J. Bosch, "Design decisions: The bridge between rationale and architecture", In A. H. Dutoit, R. McCall, I. Mistrik, and B. Paech, editors, Rationale Management in Software Engineering, Springer-Verlag, chapter 16, **(2006)** March, pp. 329–348.
[17] D. Falessi, G. Cantone, R. Kazman and R. Kruchten, "Decision making techniques for software architecture design: A Comparative survey", ACM computing surveys, vol. 43, no. 4, **(2011)**, pp. 33:1-33:28.
[18] A. Erfanian and F. S. Aliee, "An Ontology-Driven Software Architecture Evaluation Method", Proc. Workshop SHARK, ACM Computing, **(2008)**, pp. 79-86.
[19] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crub´ezy, H. Eriksson, N. F. Noy, and S. W. Tu, "The evolution of protgé: an environment for knowledge-based systems development", Int. J. Hum.-Comput. Stud., vol. 58, no. 1, **(2003)**, pp. 89–123.
[20] M. C. Daconta, K. T. Smith and L. J. Obrst, "The Semantic Web: A Guide to the future of XML, Web services, and Knowledge Management", John Wiley & Sons, Inc., New York, NY, USA, **(2003)**.
[21] A. Tang, "A rationale-based model for architecture design reasoning", Ph.D. Thesis, Faculty of ICT, Swinburne University of technology, **(2007)** February.
[22] F. Losavio, L. Chirinos, N. Levy and A. Ramdane, "Quality Characteristics of Software Architecture", Journal of Object Technology, vol. 2, no. 2, **(2003)** March, pp. 133-150.
[23] A. Volker and J. Giesen, "Requirements Interdependencies and Stakeholders Preferences", Proceedings of the 10th Anniversary IEEE Joint international Conference on Requirements Engineering, **(2002)** September, pp. 206-212.
[24] L. Zadeh, "Fuzzy sets", Information and Control, vol. 8, no. 3, **(1965)**, pp. 338-353.
[25] E. Triantaphyllou, "Multi-Criteria Decision Making: A Comparative Study", Norwell, MA: Kluwer, **(2000)**.
[26] P. J. M. Laarhoven and W. Pedrycz, "A fuzzy extension of Saaty's priority theory", Fuzzy Sets Syst., vol. 11, **(1983)**, pp. 229–241.
[27] M. H. Vahidnia, A. Alesheikh, A. Alimohammadi and A. Bassiri, "Fuzzyanalytical hierarchy process in GIS application", The InternationalArchives of the Photogrammetry, Remote Sensing and SpatialInformation Sciences, Beijing, vol. 37, **(2008)**, pp. 593-596.