A Model for Reusability of Software Architectural Knowledge

A Thesis Presented to

Department of Computer Science and Software Engineering
Faculty of Basic & Applied Sciences

In Partial Fulfillment

of the requirement for the degree

Of

Master of Sciences (Software Engineering)

By

Fawad Ahmad Khan

(109-FAS-MSSE/F-06)

International Islamic University, Islamabad

Faculty of Basic & Applied Sciences,

Department of Computer Science and Software Engineering

FINAL APPROVAL

ROJECT EVALUATION COMMITTEE

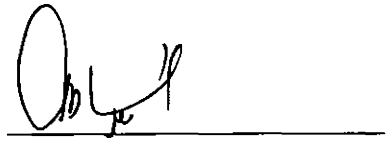
External Examiner:

Dr.Arshad Ali Shahid

Professor, Department of Computer Science,

National University of Computer & Emerging Sciences,

NU-FAST, Islamabad , Pakistan



Internal Examiner:

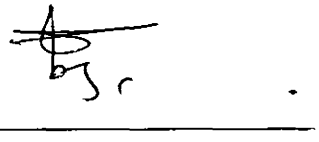
Mr.Shahbaz Ahmed Khan

Associate Professor,

Department of Computer Science and Software Engineering,

International Islamic University,

Islamabad, Pakistan.



Supervisor:

Dr.NaveedIkram

Associate Professor,

Faculty of Computing,

Riphah International University,

Islamabad, Pakistan.

A Dissertation submitted as

Partial Fulfillment of Requirements

For the degree of Master of Science in

Software Engineering

Declaration

I hereby declare and affirm that this thesis neither as a whole nor as part thereof has been copied out from any source. It is further declared that I have completed this thesis entirely on the basis of my personal effort, made under the sincere guidance of my supervisor. If any part of this report is proven to be copied out or found to be a reproduction of some other, I shall stand by the consequences. No portion of the work presented in this report has been submitted in support of an application for other degree or qualification of this or any other University or Institute of learning.

Fawad Ahmad Khan
109-FAS/MSSE/F-06

Acknowledgment

This thesis has been in many aspects the ultimate confrontation with some of my own limitations. Happily enough, I have fought and overcome them, but I couldn't have done this without the help of many people. Some of them I would like to especially acknowledge here.

First of all, I would like to thank my supervisors: **Dr.NaveedIkram** for his guidance and demonstrating to me that no research goal is too high to aim for. Mr. Muhammad Usman, for bringing reality to my vision, who initially introduced me about the state of practice in this area of research and Mr. Shahbaz Ahmad Khan for always having wise advice available when I was stuck, to Mr. Adnan Ashraf, for providing intelligent comments during finalization of proposal for this thesis.

I would also like to express my gratitude to all my MS fellows for their moral support.

Finally, to wrap-up this acknowledgement, I would like to thank my mother for her unconditional love and support during all these years.

Abstract

This research work aims at improving reusability of software architectural knowledge. In knowledge management literature software architectural knowledge is categorized as technical knowledge e.g. architecture styles, tactics and reference architecture etc and contextual knowledge of software architecture that concerns why the things are like the way they are. Software architectural decisions are based on experience, knowledge, intuition and exposure of software architect.Therefore a case study was conducted to identifycontextual knowledge elements that influence software architects in decisions of selecting appropriate software architecture knowledge elements. The outcome of conducting the case study is the identification of contextual knowledge elements that drives software architects in selection of appropriate technical knowledge element. Based on analysis of case study results a model for reusability of software architectural knowledge has been proposed.

Table of Contents

1 Introduction

1.1 Software Architecture

One sub-discipline within software engineering is software architectures, which is a kind of high-level design of the software of one or more systems. Currently, there is no agreement to what exactly software architecture involves. A review of the published literatureindicates that a unified view of software architecture has not been approaching [30]. Software architectures shiftdeveloper'sfocus from linesof-code to architectural elements and their interconnection structure [31]. One popular definition is from [13]:

> *The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.*

It is generally accepted that the constituents of the software architecture are components, connections, constraints, styles and patterns [36]. A software architecture design provides early system foundation for subsequent detailed design and implementation [37].Software architecture are created, evolved, and maintained in a complex environment. The architecture business cycle [13] of figure 1.1 illustrates this. On the left hand side, the figure presents different factors that influence software architecture through an architect. It is the responsibility of the architect to manage these factors and architecture of the system. An important factor is formed by requirements, which come from stakeholders and the developing organization.

The architecture business cycle [13] contains a feedback loop, within which the architect's influences are influenced by both the system and architecture. This feedback loop exists, since the perception of the system and the architecture influences the stakeholders.

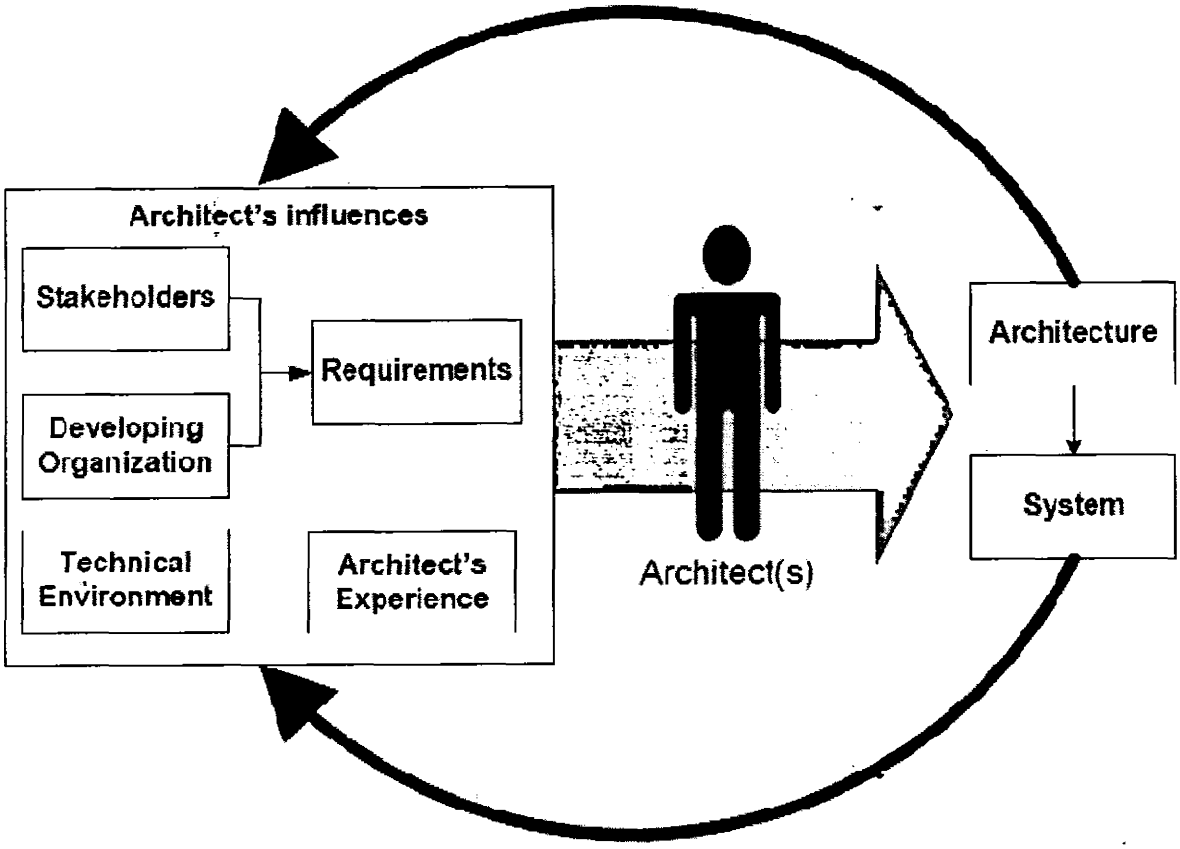


Figure 1.1: The architecture business cycle from [13]

1.2 Knowledge Management

Knowledge Management is defined as:

"*The process of selectively applying knowledge from previous experiences of decision-making to current and future decision making activities with the express purpose of improving the organization's effectiveness [27].* "

Therefore major goals of knowledge management are defined in [27] as:

- Identify Critical Knowledge
- Acquire Critical Knowledge in a Knowledge Base or Organizational Memory
- Share the stored Knowledge
- Apply the Knowledge to appropriate situations

- Determine the effectiveness of using the applied knowledge
- Adjust Knowledge use to improve effectiveness

The major purpose of knowledge management is to improve business processes and practices by utilizing individual and organizational knowledge resources. These include skills, capabilities, experiences, routines, cultural norms, and technologies [12]. Applying knowledge management techniques to project activities can improve productivity and reduce risks of failures. In the knowledge management literature, a distinction is often made between the personalization strategy and the codification strategy [1, 11,12].

1.2.1 Personalization

The personalization strategy emphasizes interaction between knowledge workers. The knowledge itself is kept by its creator.

1.2.2 Codification

In the codification strategy, the knowledge is codified and stored in a repository. The repository may be unstructured, or structured according to some model.

1.3 Software Architectural Knowledge

Architectural knowledge includes the knowledge involved with software architectures. Architectural knowledge is vital for the architecting process, as it improves the quality of this process and of the architecture [28]. What precisely the view of AK involves is still a topic of ongoing research and debate [7]. Some define "*AK as AK = design decisions + design*" [29], others as "*AK = drivers, decisions, analysis*" [8], and some take a broader perspective by including processes and people aspects [19]. Most researchers agree that at least one part of AK is about the rationale, assumptions, and context of decisions that lead to a particular design.

1.4 Classification of Software Architecture Knowledge

The knowledge can be technical (such as patterns, tactics, and quality attribute analysis models) this type of knowledge is required to identify, assess, and select suitable design options for design decisions. "*The knowledge can be contextual, also called design rationale, such as design options considered, tradeoffs made, assumptions, and design reasoning [2].*"

Similarly, types of software architecture knowledge are described in [10]. The application-generic knowledge that architects have implicitly in their heads, from their experience in working in one or more domains.This is like library of knowledge, which consists of architectural patterns, tactics or reference architectures or even other software engineering techniques, e.g. in requirements engineering. This type of knowledge can be generally applied in several applications regardless of the domain.

The application-specific knowledge, of a specific application during the initial development or evolution of that application. This involves all the decisions that were taken during the architecting process of a particular system and the architectural solutions that implemented the decisions.

"Architects require topic knowledge (learned from text books and courses) and episodic knowledge (experience with the knowledge) [12], to design software architectures."

Based on the above categorization of knowledge, software architecture knowledge can be broadly categorized as general knowledge of software architecture and knowledge which is applied in particular project during architecture design process.

1.5 Research Problem

Software architecture design is knowledge intensive process that produces and requires knowledge. Commonly during the architecture development process, decisions are not documented explicitly but are reflected by the models the architects build, consequently, useful knowledge attached to the decisions and its process is lost. The software architecture knowledge can be categorized as technical knowledge [2] (such as patterns, tactics, and quality attribute analysis models) and contextual knowledge (design rationale) [2].

Software architecture embodies significant decisions, these decisions are in the form of tacit knowledge, but rationales behind the decisions are not available. This causes two main problems:

- Design decisions vaporize.
- The reusability of technical knowledge applied in designing similar software architecture is difficult.

This research is related to the field of Software Architecture Knowledge Management and will answer the following question:

What contextual knowledge software architects require for reusability of technical knowledge of software architecture?

1.6 Research Methodology

The research method was comprised of following components in the given order:

- Detailed literature review and critical analysis was conducted, in order to identify state of practice and knowledge management approaches in software architecture knowledge management to identify characteristics, effectiveness and limitation of each.
- After detailed literature review an exploratory case study was conducted to identify contextual knowledge elements used by software architects in designing software architecture.
- Based on the results of case study a thorough analysis of results was performed in order to identify reusability needs for selection of appropriate technical knowledge elements.
- Proposed a model for reusability of softwarearchitectural knowledge.
- A prototype for proof of concept was built in order to validate results.
- Recommendations for the future work.

1.7 ResearchRationale

The outcome of this research is the identification of reusability needs in form of contextual knowledge elements that software architects require for new architectural decisions. In addition,outcome also includes a model for reusability of software architectural knowledge.

2 Literature Review

2.1 Decision Goal and Alternatives DDR Framework (DGA-DDR)

The DGA DDR framework [9]attempts to document the reason for design decision. The Decision Goal and Alternatives (DGA) DDR [9] is motivated by the decision goals and design alternative available." Decision Type describes problem to be solved and Decision Alternative (DA) corresponds to an available alternative with respect to a Decision Type (DT). "The rationale behind a design decision documents the attributes of CBAM [9]. According to DGA, whatever the software context might be, design decisions depend on basic decision goals and inter-decision relationships. The entities that influences design decision rationales are functional requirements, non-functional requirements, business goals and decision relationships [9].

2.1.1 Characteristics

This framework supports in making new decisions as the framework decomposes the concept of decision into Decision Type (DT) and Decision Alternative (DA) [9].The framework improved design quality and reusability and also explicit domain knowledge [9]is available to be used by other employees. The framework exploits value based-principles for use of systematic use of design decision rationale [9].The framework analyzes scenarios on the basis of context, matrix and required design decision information [9].

2.1.2 Limitations

1. The framework does not provide any type of tool support.
2. The framework does not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
3. The framework does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
4. The framework does not associate classification of non-functional requirements with technical knowledge elements and contextual elements.

2.2 A Core Model of Architectural Knowledge

The main aim of core model [19] of AK is to give a common frame of reference for architectural knowledge sharing [19]. This research work proposes a model of architectural knowledge that has maximal expressivity in the architectural knowledge domain and functions as a reference model for sharing architectural knowledge [19].

2.2.1 Characteristics

The aim of this research work and model is to provide minimalistic set of vocabulary that describes software AK [19]. This research work helps improves knowledge management practices as it recordsoptions and selected option by proposing and ranking [19]. The software knowledge management applications are analyzed regarding four perspectives of sharing, compliance, discovery and traceability [19].

2.2.2 Limitation

1. The core model does not provide any type of tool support [19].
2. The core model does not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
3. The core model does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
4. The core model does not associate classification of non functional requirements with technical knowledge elements and contextual elements.

2.3 Variability Modeling Principles to Capture Architectural Knowledge (COVAMOF Framework)

COVAMOF [18] is a variability modeling framework that consists of models, tooling and processes that support engineers in the development of product families in addition to the configuration of individual products from a product family [18]. The vision is use variability modeling philosophy to storeAK [18]. The idea behind COVAMOF is that it gives several views on the variability that is presented by the product family artifacts [18].

2.3.1 Characteristics

The proposal behind COVAMOF framework [18] exploitation in software architecture knowledge management is that it provides several views on the variability that is provided by the product family artifacts [18].The framework deals with the imprecise and incomplete nature of the effect of decisions on quality attributes [18].It enables tool support to manage complexity of software architecture design activity [18].

2.3.2 Limitations

1. The COVAMOF framework does not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
2. The COVAMOF framework does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
3. The COVAMOF framework does not associate classification of non functional requirements with technical knowledge elements and contextual elements.

2.4 Evolution Tailored with Architectural Knowledge (ETAK)

Evolution Tailored with Architectural Knowledge), ETAK [38] approach of software architecture knowledge management facilities the software architect for software evolutions needs. The architect can specify properties to be considered, which are described as traits for inclusion in the evolution. ETAK investigate the architectural knowledge to ascertain relevancy, facilitating the software architect in deciding whether traits are important for the architecture [38].

2.4.1 Characteristics

ETAK examines the architectural knowledge to determine the relevance of evolutions traits required for software architecture design process. Using ETAK software architect can adjust a number of inputs, like traits, scope, and architectural knowledge and process again ETAK. ETAK enables software architect to leverage, according to relevance results and/or tailored evolution attained.

2.4.2 Limitations

1. The framework does not provide any type of tool support [38].
2. The framework is mainly targets software architect needs to address software architecture evolution needs.
3. The framework does not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
4. The framework does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
5. The framework does not associate classification of non functional requirements with technical knowledge elements and contextual elements.

2.5 Process-centric Architecture Knowledge Management Environment (PAKME)

The framework is founded onnotion from knowledge management, experience factory, andpattern-mining [12]". It consists of various approaches to capture design decisions and contextual information. The knowledge repository is logically divided into knowledge-based artifacts, generic knowledge, and project-based artifacts [12]. "PAKME is composed of four components [12]; knowledge acquisition, knowledge maintenance, knowledge retrieval, and knowledge presentation [12].

2.5.1 Characteristics

This framework entails an approach to document architectural information from patterns, and engage a data model to explain architectural constructs, their attributes and relationships [12]. It provides support for design and analysis methods [12]. The frameworkprovides limited reusability of technical knowledge of software architecture independent of contextual factors [12].

2.5.2 Limitation

1. As software architecture design is an iterative process this framework does not supports such nature of software architecture design. This result in loss of valuable knowledge of why the particular decision is changed and its corresponding rationales.
2. This tool is not open source and also lacks groupware support.
3. It caters the need for basic reusability from software architecture knowledge repository besides performing other common tasks of knowledge management.
4. The framework does not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
5. The framework does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
6. The framework does not associate classification of non functional requirements with technical knowledge elements and contextual elements.

2.6 Architecture Design Decision Support System (ADDSS)

ADDSS [21] is a web-based tool for storing architectural design decisions. ADDSS makes the architecture by iterative process where one or more design decisions are made for each of the iterations [21]. This tool is founded on the meta-model for software architecture knowledge management [21].

2.6.1 Characteristics

ADDSS tools supports gradual formalization i.e helps in learning[21]. It enables multi-perspective support for different stakeholders. Further, this tool is an open source tool [21]. It allows the storage of several projects and architectures [21]. It enables multi-perspective support for different stakeholders [21], as differenttypes of users with different roles (e.g.: project managers, architects, etc.) can be registered by filling a simple form and the system emails them a username and a password. The meta-model of ADDSS relate influence of assumptions in decisions [21].The meta-model of ADDSS consider affects of contextual factors on technical knowledge elements [21].

2.6.2 Limitation

1. The meta-model of ADDSS does not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
2. The meta-model of ADDSS does not relate influences of contextual factors considered as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
3. The meta-model of ADDSS does not associate classification of non functional requirements with technical knowledge elements and contextual elements.
4. The tool is not tested in an industrial setting [21].

2.7 ARCHIUM

The Archium tool [17] is a prototype implementation of the knowledge grid presented in the Griffin project [1, 17]. A meta-model has been defined by Archium which is build from three sub-models [17]: an architectural model, a design decision model, and a composition model which compose design fragments (an architectural fragment defining a collection of architectural entities). The prototype contains a compiler and a supportive run-time environment [22].

2.7.1 Characteristics

The Archiumunite an architectural description language (ADL) with Java language [17] to express the elements from a component and connector view and making precise the design decisions and its rationale."*This includes a code transformation process, which analyzes the architectural elements and transforminto Java classes."[17]*This makes sure implementation to design consistency.

2.7.2 Limitation

1. The Archium tool has not been tested yet in an industrial setting [17].
2. As the tool employs ADL [17], with Java, so its applicability for a knowledge management tool is limited because of underlying ADL.
3. Since tool supports component and connector view so it lacks multi perspective and iteration support [21].

4. This tool is more applicable in environments where architects are more experienced and development environment is java based. The reusability of knowledge is limited. And also this tool is not open source.
5. The tool does not provide any type of tool support[17].
6. The tool does not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
7. The tool does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
8. The tool does not associate classification of non functional requirements with technical knowledge elements and contextual elements.

2.8 AQUA

AQUA [14]is an approach for decision centric architecture designwhich is based on the proposed model. The proposed model represents architectural design decisions for building architectural design decisions clear [14]. AQUA defines decision-centric [14] process of finding, evaluating, and changing the decisions. During the decision-centric process, the AQUA involved works of architectural evaluation and transformation [14].

2.8.1 Characteristics

"The integrated approach AQUA [14] supports finding, analyzing and changing decisions. It supports architects in evaluation phase.AQUA integrated the activities relevant to quality achievement at the architectural level, which include architectural evaluationand transformation [14]."

2.8.2 Limitation

1. AQUA [14] includes important concept but theeffort for facilitationis more focused to help architect in software architecture evaluation phase.
2. AQUA[14] does not provide any type of tool support.

3. The AQUAdoes not relate decisions with contextual factors influencing the decision. Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
4. The AQUA 'does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
5. The AQUAdoes not associate classification of non functional requirements with technical knowledge elements and contextual elements.

2.9 Automatic Architecture Knowledge Extraction Tool (AAKET)

AAKET [16] is tool that collects architectural knowledge from documents and electronic mails and records it in structured manner in knowledge repositories, by minimum user intervention [16]. The research goal for AAKET is to achieve an appropriate mean of capturing architectural knowledge, and transforms this knowledge into software architecture knowledge management tools [16]. AAKET addresses the knowledge management issues like structured information is not available and lack of motivation to put efforts for knowledge management [16]. The main focus is on reducing the effort to capture design decisions. To achieve performance up to desired level AAKET is developed using Visual C++ 6.0 [16]. In addition, AAKET employed best algorithms to address performance issues [16].

2.9.1 Characteristics

AAKET address the issues of manually transferring architecture knowledge from documents to knowledge repositories [16]. AAKET perform most of the lengthy and laborious tasks semi-automatically with minimum human intervention [16]. It manages the authentication by storing all the authentication related data in a remote machine [16]. It extracts the information stored in electronic mails and other documents based on a set of rules, and hands it over to the next layer component for persistence of knowledge. [16]. AAKET uses PAKME [20] (Process-based architecture knowledge management environment) as knowledge repository [16].

2.9.2 Limitation

1. Although this tool minimize the effort required for storing and managing software architecture knowledge but currently its utility is limited in terms of features and also organizations store their documents and artifacts in different file formats but this tool is currently good for MS word and MS-outlook [16].
2. This tool extracts architectural knowledge so it is more applicable where need is to share architecture knowledge among various stakeholders. This tool is more applicable for managing post architecture knowledge for sharing purposes.
3. This tool is not yet used by industry practitioners [16].
4. This tool is not open source and also lacks groupware support [16].
5. The tool does not relate decisions with contextual factors influencing the decision.Therefore rationales for the decisions are not structured resulting limited reuse of existing decisions.
6. The tool does not relate influences of contextual factors considered as assumption or as constraints on selection of specific technical knowledge element i.e. architecture style of tactics. Hence maintaining knowledge in such a way only benefits for the life cycle of current project.
7. The tool does not associate classification of non functional requirements with technical knowledge elements and contextual elements.
8. Currently this tool only stores its information to repository of PAKME [16].

2.10 Summary of Survey Techniques

No	Technique	Characteristics	Limitations
1	Decision Goal and Alternatives DDR Framework (DGA-DDR)	• Decomposes the concept of decision into Decision Type (DT) and Decision Alternative (DA). • The framework exploits value based-principles for use of systematic use of design decision rationale	• No tool support. • Not relate decisions with contextual factors. • Not relate influences of contextual factors considered as assumption or as constraints.

2	A Core Model of Architectural Knowledge	• Provide minimalistic set of vocabulary. • Consider four perspectives of sharing, compliance, discovery and traceability of software architecture.	• No tool support. • Not relate decisions with contextual factors. • Not relate influences of contextual factors considered as assumption or as constraints.
3	Variability Modeling Principles to Capture Architectural Knowledge (COVAMOF Framework)	• Support in the development of product families in addition to the configuration of individual products from a product family using variability modeling.	• Does not relate decisions with contextual factors influencing the decision. • Does not relate influences of contextual factors considered as assumption
4	Evolution Tailored with Architectural Knowledge (ETAK)	• Facilities the software architect for software evolutions needs. • Examines the architectural knowledge to determine the relevance of evolutions traits required for software architecture design process: • Enables architect to leverage, according to relevance results and/or tailored evolution attained.	• No tool support. • Not relate decisions with contextual factors. • Not relate influences of contextual factors considered as assumption or as constraints.

5	Process-centric Architecture Knowledge Management Environment (PAKME)	• Founded on notion from knowledge management, experience factory, and pattern-mining. • Consists of various approaches to capture design decisions and contextual information. • Logically divided into knowledge-based artifacts, generic knowledge, and project-based artifacts.	• Does not support iterative process this framework. • This tool is not open source and also lacks groupware support. • Does not relate decisions with contextual factors influencing the decision. • Not relate decisions with contextual factors. • Not relate influences of contextual factors considered as assumption or as constraints.
6	Architecture Design Decision Support System (ADDSS)	• Web-based tool for storing architectural design decisions. • ADDSS makes the architecture by iterative process where one or more design decisions are made for each of the iterations • Founded on the meta-model for software architecture knowledge management.	• The meta-model of ADDSS does not relate decisions with contextual factors influencing the decision. • Does not relate influences of contextual factors considered as constraints. • Does not associate classification of non-

			functional requirements with technical knowledge elements and contextual elements.
7	ARCHIUM	• Unite an architectural description language (ADL) with Java. • This includes a code transformation process, which analyzes the architectural elements and transform into Java classes."[17] This makes sure implementation to design consistency.	
8	AQUA	• Supports finding, analyzing and changing decisions. • Supports architects in evaluation phase.	• Not relate decisions with contextual factors. • Not relate influences of contextual factors considered as assumption or as constraints.
9	AAKET	• Perform most of the lengthy and laborious tasks semi-automatically with minimum human intervention. • Extracts the information stored in electronic mails and other documents based on a	• No tool support. • Not relate decisions with contextual factors. • Not relate influences of contextual factors considered as assumption or as constraints.

		set of rules, and hands it over to the next layer component for persistence of knowledge.	

3 Case Study Design

This research is intended to identify the contextual knowledge elements used by software architects during software architecture design, so as we address their reusability needs during architecture design.The primary function of identification is to improve reusability of software architecture technical knowledge based on contextual knowledge as there is growing interest in methods for capturing the rationale behind software architectures [34]. Case Study research is an ideal methodology when a holistic and in depth analysis is required for such situation. In our study we want to identifycontextual knowledge elements that determine the selection of particular technical knowledge element.

3.1 What is Case Study?

The analytical research is not adequate for investigating difficult real life issues, involving humans and their interactions with technology [23].The case study gives the story behind the result by capturing what happened to bring it about, and can be a good opportunity to highlight a project's success, or to bring attention to a particular challenge or difficulty in a project [33]. Case study is a suitable research methodology for software engineering research since it studies contemporary phenomena in its natural context [23].

3.2 Rationale for Selection of Case Study as a Research Methodology

The rationale for selection of case study as a research methodology for this research was to explore designing of software architecture design process in terms of what guides selection of decisions, and factors that determine reusability needs of software architects.The main objective of this research thesis is identification of software architectural knowledge elements that influences the software architects in determining the right technical knowledge element in a given context. As software architecture design phenomenon takes place for a given industrial project or system having definite customer, sponsor's requirements or expectations including certain assumptions and constraints on group of software architects designing software architecture. Therefore it was notappropriate to develop any environment in lab settings where industry software architects design software architecture for some real industry project with definite requirements and constraints, so experiment [39] research methodology was not

considered for this research work. In addition, this research lacks any preliminary hypothesis required to conduct experiment.

When it is hard to experiment due to high cost, complexity, and inconvenienceor impossible to experiment due to any reason, another strong candidate i.e. simulation[40]research methodology was considered. Since in this research workit was inconvenient to collect a required group of people i.e. software architects and impossible as to simulate environment for not real requirements, assumptions and constraints. Therefore simulation is not considered for this research.

The other strong candidate research methodology considered for this research was to conduct a survey[41] because the survey is a non-experimental, descriptive research method.Conducting surveys can be useful when data needs to be collected on phenomena that cannot be directly observed.Since the subject of investigation for this research was software architecture design process and we wanted to gain a deep understanding of this phenomenon to identify right contextual knowledge elements used. Therefore conducting survey for identification of reusable software architectural contextual knowledge elements was ruled out.

3.3 Objective of Case Study:

To determine reusability needs of software architecture technical knowledge. The case study for the research is an exploratory case study. The unit of analysis for this case study is software project.

3.4 The Case:

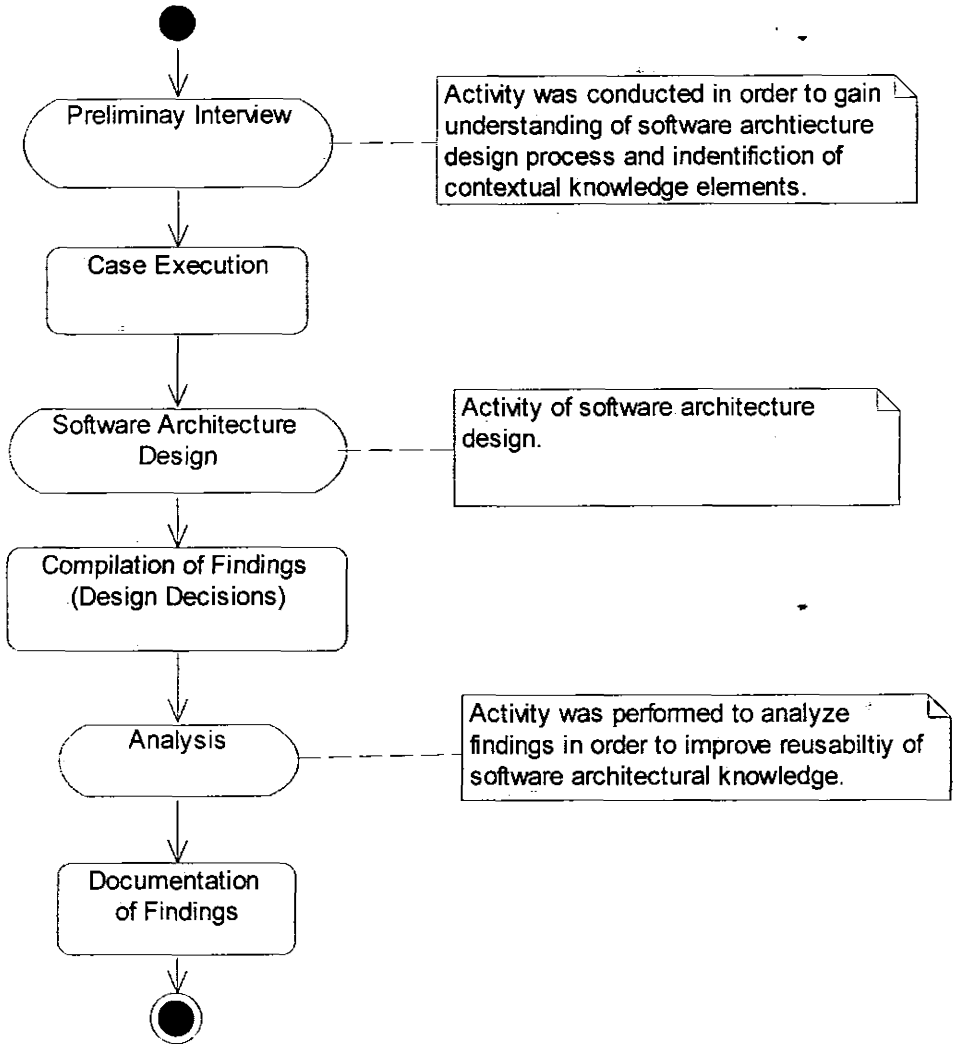
The case for this case study is software architecture design process.

3.5 Method (Data Collection):

Preliminary interview of participants were conducted to elicit contextual knowledge elements of software architectural knowledge. Comprehensive questionnaire, attached in appendix A, was evolved during the case study and at the end of case study participants filled that questionnaire. On the basis of case study results, a thorough analysis was performed for identification of reusable technical knowledge constructs.

3.6 Main Research Activities

The diagram depicts the research methodology activities.



4 Identification of Contextual Knowledge Elements

This research is intended to identify the contextual knowledge elements used by software architects during software architecture design and determining how to preserve contextual elements to enhance reusability of software architectural knowledge. This section also describes the findings of case study in terms of design decisions taken during case study.

4.1 Findingsof Case Study

Following are the software architecture contextual knowledge elementsidentified [32].

4.1.1 Application Type

This contextual element determines type of application based on requirements and infrastructure limitation. For example web application, mobile application, rich internet client, real time application etc. During software architecture design this contextual element is either assumption or constraint for software architect.

4.1.2 Time

This contextual element concerns about development and maintenance time.Constraints on time arepre-defined time linesfrom stakeholders and assumptions on time are considered as supposition of available time for software project.

4.1.3 Software Process

This contextual element describes software process followed during life cycle of application. Constraints on software architect regarding software process are to follow certain software process e.g. water fall, iterative or incremental etc for the underlying application in architecture design. An assumption regarding software process during software architecture design refers supposition of software process to be followed.

4.1.4 Implementation Technology

This contextual element describes tools and technologies catering all development needs of software engineers, for example Java, .Net, Oracle, or open source technologies. Constraints on software architecture regarding implementation technology are to follow certain technology. An

assumption regardingimplementation technologyduring software architecture design refers supposition of implementation technologyto be used.

4.1.5 Deployment Environment

This contextual element describes target environment in terms of platform and hardware for example, sun oracle machines, Intel family servers and consideration of operating system as Microsoft, UNIX or Linux. Constraints regarding deployment environment are to go with certain hardware platform and operating system. An assumption regardingdeployment environmentdüring software architecture design refers supposition of certain deployment environment.

4.1.6 Project Development Team

This contextual element describes experience, skill set and number of team members of software development project. Constraints regarding project development team are team with specific skill for example java team will be free after at time of development so constraint is project development team. Assumptions on project development team are suppositions of software architect regarding experience and skill set of team members responsible for development of software.

4.1.7 Organization Processes

This contextual element describes development organization processes for example ISO, CMMI and SPICE etc. Constraints are specific organization process say CMMI to be followed. An assumption regardingorganization processesduring software architecture design refers supposition of certain organization processes.

4.1.8 Global Software Development

This contextual element describes whether software development activities for the underlying project to be conducted collocated or in distributed settings. Constraints on software architect are either collocated or in distributed settings. Assumptions on global software development are the supposition of software architect for either mentioned possibility.

4.1.9 Cost

This contextual element describe available project budget. Constraints on cost for software architect regarding cost are either high or low cost. Assumptions on cost are the supposition of software architect for either mentioned possibility.

4.1.10 Stakeholders

This contextual element describes different people influence software architecture design other than software architecture design group.

4.1.11 Deployment Topologies

This contextual element describes target environment in terms network arrangement for example wired or wireless or star bus, ring topology. Constraints regarding deployment environment are to go with certain topology. Assumptions on deployment environment are the supposition of software architect regarding target environment network topologies.

4.1.12 Application Domain

This contextual element describes domain of application as financial, health care, telecommunications, defense or any generic component. Constraint regarding application domain is clear understating regarding domain. Assumptions on application domain are the supposition of software architect, which otherwise means not clarity of domain of application to be developed, for example development of certain general purposecomponent.

4.2 Finding Details

4.2.1 Application Typeas Assumption or Constraint

4.2.1.1 Application Type as Consideration

Influenceon Selection of Architecture Style

- Participant A considered application type consideration as medium for product requirement, high for organization requirement and medium for external requirement.
- Participant B considered application type consideration as medium for product requirement,organization requirementand external requirement.

Influence on Selection of Tactics

- Participant A considered application type consideration as high for product requirement and organization requirement and medium for external requirement.
- Participant B considered application type consideration as high for product requirement and organization requirement and medium for external requirement.

Influence on Reference Architecture

- Participant A considered application type consideration as high for product requirement, medium for organization requirement and high for external requirement.
- Participant B considered application type consideration as medium for product requirement, organization requirement and external requirement.

4.2.1.2 Application Type as Constraint

Influence on Selection of Architecture Style

- Participant A considered application type as constraint as high for product requirement, organization requirement and external requirement.
- Participant B considered application type as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered application type as constraint as high for product requirement, organization requirement, and external requirement.
- Participant B considered application type as constraint as high for product requirement and organization requirement and medium for external requirement.

Influence on Selection Reference Architecture

- Participant A considered application type as constraint as medium for product requirements, organization requirements, and external requirement.
- Participant B consideredapplication type as constraint as high for product requirements, medium for organization requirements and high for external requirements.

4.2.1.3 Application Type as Assumption

Influence on Selection of Architecture Style

- Participant A considered application type as assumption as high for product requirements, and low for organization requirements and external requirement.
- Participant B considered application type consideration as low for product requirement and medium for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered application type as assumption as high for product requirements and low for organization requirements and external requirement.

- Participant B considered application type consideration as low for product requirement and medium for organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered application type as assumption as lowfor product requirements, organization requirements and external requirement.
- Participant B considered application type as assumption as medium for product requirements, organization requirements and external requirement.

Conclusion

Application type influences in selection of any software architectural decision. This contextual element is consideredin both types of influences i.e. as an assumption and constraint. Since there exist a likelihood of considering certain application type as a generic product, therefore participants strongly consider application type in design decisions during software architecture design.

4.2.2 Time as Assumption or Constraint

4.2.2.1 Time as Consideration

Influence on Selection of Architecture Style

- Participant A considered time as medium for product requirement, high for organization requirement and low for external requirement.
- Participant B considered time as high for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered time as medium for product requirement and organization requirement and low for external requirement.
- Participant B considered time as high for product requirement and organization requirement and medium for external requirement.

Influence on Reference Architecture

- Participant A considered time as medium for product requirement and organization requirement and low for external requirement.

- Participant B considered time as high for product requirement, organization requirement and external requirement.

4.2.2.2 Time as Constraint

Influence on Selection of Architecture Style

- Participant A considered time as constraint as high for product requirement, organization requirement and external requirement.
- Participant B considered time as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered timeconstraint as high for product requirement, organization requirement and external requirement.
- Participant B considered time constraint as medium for product requirement, organization requirement and external requirement.

`Influence on Reference Architecture

- Participant A considered time constraint as medium for product requirements, high for organization requirements and medium external requirement.
- Participant B considered time as constraint as mediumfor product requirements and organization requirements and high for external requirements.

4.2.2.3 Time as Assumption

Influence on Selection of Architecture Style

- Participant A considered time as assumption as medium for product requirements, medium for organization requirements and low for external requirement.
- Participant B considered time as assumption as low for product requirement and medium for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered time as assumption as high for product requirements, organization requirements and medium for external requirement.
- Participant B considered time as assumption as high for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered time as assumption as medium for product requirements, organization requirements and external requirement.
- Participant B considered time as assumption as low for product requirements, and medium for organization requirements and external requirement.

Conclusion

Time influences in selection of any software architectural decision. There is consensus of both participants that value of time is mostly known at the time of decision but if value of time is not known or will be known after software architectural decisions, in both cases time influences software architect in any software architectural decision. The rationale for consideration is significance of time factor in activity definition, activity sequencing and in effort estimation.

4.2.3 Software Process as Assumption or Constraint

4.2.3.1 Software Process as Consideration

Influence on Selection of Architecture Style

- Participant A considered software process consideration as low for product requirement, and medium for organization requirement and external requirement.
- Participant B considered software process consideration as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered software process consideration as high for product requirement, medium for organization requirement and high for external requirement.
- Participant B considered software process consideration as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered software process consideration as medium for product requirement and organization requirement and high for external requirement.
- Participant B considered software process consideration as high for product requirement and medium for organization requirement and external requirement.

4.2.3.2 Software Process as Constraint

Influence on Selection of Architecture Style

- Participant A considered software process as constraint as mediumfor product requirement, organization requirement and external requirement.
- Participant B considered software process as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered software process as constraint as medium for product requirement, organization requirement and high for external requirement.
- Participant B considered software process as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered software process as constraint as medium for product requirements and organization requirements and high forexternal requirement.
- Participant B considered software process as constraint as medium for product requirements and organization requirements and high for external requirement.

4.2.3.3 Software Process as Assumption

Influence on Selection of Architecture Style

- Participant A considered software process as assumption as medium for product requirements, organization requirements and external requirement.
- Participant B considered software process consideration as medium for product requirement, low for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered software processas assumption as medium for product requirements, and high for organization requirements and external requirement.
- Participant B considered software process consideration as low for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered software process as assumption as low for product requirements, medium for organization requirements and low for external requirement.

- Participant B considered software processas assumption as mediumfor product requirementsand low for organization requirements and external requirement.

Conclusion

Software process influences in selection of any software architectural decision.The rationale for consideration of software process in software architectural decisions is to devise proper iteration planning and effective management of development and deployment phases.

4.2.4 Implementation Technology as Assumption or Constraint

4.2.4.1 Implementation Technology as Consideration

Influence on Selection of Architecture Style

- Participant A considered implementation technology consideration as medium for product requirement, high for organization requirement and medium for external requirement.
- Participant B considered implementation technology consideration as high for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered implementation technology consideration as low for product requirement and medium for organization requirementfor external requirement.
- Participant B considered implementation technology consideration as high for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered implementation technology consideration as high for product requirement, medium for organization requirement and high for external requirement.
- Participant B considered implementation technology consideration as high for product requirement, organization requirement and external requirement.

4.2.4.2 Implementation Technology as Constraint

Influence on Selection of Architecture Style

- Participant A considered implementation technology as constraint as high for product requirement and medium for organization requirement and external requirement.

- Participant B considered implementation technology as constraint as high for product requirement and medium for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered implementation technology as constraint as high for product requirement, organization requirement, and external requirement.
- Participant B considered implementation technology as constraint as high for product requirement and organization requirement and medium for external requirement.

Influence on Reference Architecture

- Participant A considered implementation technology as constraint as medium for product requirements high for organization requirements and medium for external requirement.
- Participant B considered implementation technology as constraint as high for product requirements, medium for organization requirements and external requirements.

4.2.4.3 Implementation Technology as Assumption

Influence on Selection of Architecture Style

- Participant A considered implementation technology as assumption as high for product requirements, medium for organization requirements and low for external requirement.
- Participant B considered implementation technology consideration as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered implementation technology as assumption as high for product requirementsand organization requirements and low for external requirement.
- Participant B considered implementation technology consideration as high for product requirement and medium for organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered implementation technology as assumption as mediumfor product requirements, low for organization requirements and medium for external requirement.
- Participant B considered implementation technology as assumption as medium for product requirements, organization requirements and external requirement.

Conclusion

Implementation technology influences in selection of software architectural decision. The rationale for consideration is due to availability of certain programming language constructs like object orientation, memory management and multithreading etcrequired to efficiently implementing particular architecture style, tactics and reference architecture.

4.2.5 Deployment Environment as Assumption or Constraint

4.2.5.1 Deployment Environment as Consideration

Influence on Selection of Architecture Style

- Participant A considered deployment environment consideration as medium for product requirement, high for organization requirement and external requirement.
- Participant B considered deployment environment consideration as medium for product requirement, organization requirement and high for external requirement.

Influence on Selection of Tactics

- Participant A considered deployment environment consideration as high for product requirement, medium for organization requirement and high for external requirement.
- Participant B considered deployment environment consideration as medium for product requirement and organization requirement and low for external requirement.

Conclusion

1. Deployment environmentis considered in selection of tactics for product requirements.
2. Deployment environmentis considered in selection of tactics style for organization requirements.
3. Deployment environmentis considered in selection of tactics for external requirements.Despite disagreement among participants the rationale for consideration is due to likelihood of considering application as a generic product and goal is to achieve platform independence.

Influence on Reference Architecture

- Participant A considered deployment environment consideration as medium for product requirement, high for organization requirement and medium for external requirement.
- Participant B considered deployment environment consideration as high for product requirement, medium for organization requirement and high external requirement.

4.2.5.2 Deployment Environment as Constraint

Influence on Selection of Architecture Style

- Participant A considered deployment environmentas constraint as high for product requirement, organization requirement andexternal requirement.
- Participant B considered deployment environmentas constraint as low for product requirement and medium for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered deployment environment as constraint as high for product requirement, organization requirement and external requirement.
- Participant B considered deployment environment as constraint as low for product requirement, high for organization requirement and medium for external requirement.

Influence on Reference Architecture

- Participant A considered deployment environmentas constraint as medium for product requirements andorganization requirements and high for external requirement.
- Participant B considered deployment environmentas constraint as low for product requirements medium for organization requirements and external requirements.

4.2.5.3 Deployment Environment as Assumption

Influence on Selection of Architecture Style

- Participant A considered deployment environmentas assumption as medium for product requirements, and high for organization requirements and low for external requirement.
- Participant B considered deployment environmentconsideration as medium for product requirement, organization requirement andexternal requirement.

Influence on Selection of Tactics

- Participant A considered application type as assumption as high for product requirements, and low for organization requirements and external requirement.
- Participant B considered application type consideration as medium for product requirement and medium for organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered deployment environmentas assumption as mediumfor product requirements and low for organization requirements and external requirement.

- Participant B considered deployment environmentas assumption as mediumfor product requirements low for organization requirements and medium for external requirement.

Conclusion

Deployment environmentinfluences in selection of software architectural decision. The rationale for consideration is due to exploitation of certain available features exhibited by certain deployment environment which otherwise needs to be developed like user management or access control from active directories rather custom build user management system.

4.2.6 Organization Processes as Assumption or Constraint

4.2.6.1 Organization Processes as Consideration

Influence on Selection of Architecture Style

- Participant A considered organization processes consideration as medium for product requirement, low for organization requirement and medium for external requirement.
- Participant B considered organization processes consideration as low for product requirement, medium for organization requirement and low for external requirement.

Influence on Selection of Tactics

- Participant A considered organization processes consideration as high for product requirement, low for organization requirement and high for external requirement.
- Participant A considered organization processes consideration as medium for product requirement, low for organization requirement and high for medium requirement.

Influence on Reference Architecture

- Participant A considered organization processes consideration as medium for product requirement, high for organization requirement and external requirement.
- Participant B considered organization processes consideration as low for product requirement, medium for organization requirement and low for external requirement.

4.2.6.2 Organization Processesas Constraint

Influence on Selection of Architecture Style

- Participant A considered organization processesas constraint as medium for product requirement, organization requirement and external requirement.

- Participant B considered organization processesas constraint as medium for product requirement, organization requirement and high for external requirement.

Influence on Selection of Tactics

- Participant A considered organization processes as constraint as medium for product requirement, organization requirement and external requirement.
- Participant B considered organization processesas constraint as low for product requirement and organization requirement and medium for external requirement.

Influence on Reference Architecture

- Participant A considered organization processesas constraint as high for product requirements and medium for organization requirements and external requirement.
- Participant B considered organization processesas constraint as medium for product requirements and organization requirements and high for external requirements.

4.2.6.3 Organization Processesas Assumption

Influence on Selection of Architecture Style

- Participant A considered organization processesas assumption as low for product requirements, organization requirements and external requirement.
- Participant A considered organization processes as assumption as medium for product requirements, organization requirements and external requirement.

Influence on Selection of Tactics

- Participant A considered organization processesas assumption as low for product requirements, and high for organization requirements and low for external requirement.
- Participant B considered organization processesconsideration as low for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered organization processesas assumption as low for product requirements, medium for organization requirements and low for external requirement.
- Participant B considered organization processesas assumption as medium for product requirements, organization requirements and external requirement.

Conclusion

Organization processes influences in selection of software architectural decision. The rationale for consideration is because organization processes are mostly concerned with what to achieve rather how to accomplish the objectives. As quality software product is aim of organization so participants considerthis factor influences in all decisions regardless of software architectural decision or any other decision in life cycle of product.

4.2.7 Project Development Team as Assumption or Constraint

4.2.7.1 Project Development Team as Consideration

Influence on Selection of Architecture Style

- Participant A considered project development team consideration as medium for product requirement, organization requirement and high for external requirement.
- Participant B considered project development team consideration as medium for product requirement, organization requirement and low for external requirement.

Influence on Selection of Tactics

- Participant A considered project development team consideration as high for product requirement and organization requirement and low for external requirement.
- Participant B considered project development team consideration as low for product requirement, medium for organization requirement and low for external requirement.

Influence on Reference Architecture

- Participant A considered project development team consideration as medium for product requirement, low for organization requirement and medium for external requirement.
- Participant B considered project development team consideration as medium for product requirement, organization requirement and low for external requirement.

4.2.7.2 Project Development Team as Constraint

Influence on Selection of Architecture Style

- Participant A considered project development team as constraint as high for product requirement, organization requirement and external requirement.
- Participant B considered project development team as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered project development team as constraint as high for product requirement, organization requirement andexternal requirement.
- Participant B considered project development team as constraint as medium for product requirement and organization requirement and medium for external requirement.

Influence on Reference Architecture

- Participant A considered project development team as constraint as medium for product requirements, organization requirements, and external requirement.
- Participant B considered project development team as constraint as medium for product requirements, organization requirements, and external requirement.

  The rationale for consideration is in case of constraints on team members application of reference architecture decision will be on the basis of skill set and capabilities of team members.

4.2.7.3 Project Development Team as Assumption

Influence on Selection of Architecture Style

- Participant A considered project development team as assumption as medium for product requirements and low for organization requirements and high for external requirement.
- Participant B considered project development team consideration as low for product requirement and medium for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered project development team as assumption as high for product requirements, organization requirements and external requirement.
- Participant B considered project development team consideration as low for product requirement and medium for organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered project development team as assumption as medium for product requirements and organization requirements and high for external requirement.
- Participant B considered project development team as assumption as medium for product requirements, organization requirements and external requirement.

Conclusion

Project development team influences in selection of software architectural decision. The rationale for consideration is due to skill set and experience requirements in certain implementation technologies in order to develop quality product.

4.2.8 Global Software Development as Assumption or Constraint

4.2.8.1 Global Software Development as Consideration

Influence on Selection of Architecture Style

- Participant A considered global software development consideration as medium for product requirement, high for organization requirement and external requirement.
- Participant B considered global software development consideration as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered global software development consideration as high for product requirement, medium for organization requirement and low for external requirement.
- Participant B considered global software development consideration as medium for product requirement, organization requirement and external requirement.

The rationale for consideration in all requirements is to effective management of project life cycle as selection of appropriate tactics helps in effort distribution in global settings.

Influence on Reference Architecture

- Participant A considered global software development consideration as medium for product requirement, high for organization requirement and medium for external requirement.
- Participant B considered global software development consideration as medium for product requirement, high for organization requirement and medium for external requirement.

4.2.8.2 Global software development as Constraint

Influence on Selection of Architecture Style

- Participant A considered global software development as constraint as high for product requirement and organization requirement and medium for external requirement.

- Participant B considered global software development as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered global software development as constraint as medium for product requirement and highfor organization requirement and external requirement.
- Participant B considered global software development as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered global software development as constraint as medium for product requirements low for organization requirements and high for external requirement.
- Participant B considered global software development as constraint as medium for product requirements, organization requirements and external requirements.

4.2.8.3 Global software development as Assumption

Influence on Selection of Architecture Style

- Participant A considered global software development as assumption as medium for product requirements, and low for organization requirements and medium for external requirement.
- Participant B considered global software development consideration as medium for product requirement,organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered global software development as assumption as low for product requirements, medium for organization requirements and external requirement.
- Participant B considered global software development consideration as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered global software development as assumption as mediumfor product requirements low for organization requirements and external requirement.

- Participant B considered global software development as assumption as low for product requirements and medium for organization requirements and external requirement.

Conclusion

Global software development influences in selection of software architectural decision.

4.2.9 Cost as Assumption or Constraint

4.2.9.1 Cost as Consideration

Influence on Selection of Architecture Style

- Participant A considered cost consideration as high for product requirement, medium for organization requirement and external requirement.
- Participant B considered cost consideration as medium for product requirement, high for organization requirement and medium for external requirement.

Influence on Selection of Tactics

- Participant A considered cost consideration as high for product requirement and organization requirement and low for external requirement.
- Participant B considered cost consideration as high for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered cost consideration as medium for product requirement and organization requirement and high for external requirement.
- Participant B considered cost consideration as high for product requirement and organization requirement and medium for external requirement.

4.2.9.2 Cost as Constraint

Influence on Selection of Architecture Style

- Participant A considered cost as constraint as high for product requirement and organization requirement and medium for external requirement.
- Participant B considered cost as constraint as medium for product requirement and organization requirement and medium for external requirement.

Influence on Selection of Tactics

- Participant A considered cost as constraint as high for product requirement, medium for organization requirement and high for external requirement.
- Participant B considered cost as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered cost as constraint as high for product requirements, low for organization requirements and high for external requirement.
- Participant B considered cost as constraint as medium for product requirement, organization requirement and external requirement.

4.2.9.3 Cost as Assumption

Influence on Selection of Architecture Style

- Participant A considered cost as assumption as medium for product requirements, low for organization requirements and medium for external requirement.
- Participant B considered cost as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered cost as assumption as low for product requirementsand medium for organization requirements and external requirement.
- Participant B considered cost as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered cost as assumption as mediumfor product requirements and low for organization requirements and external requirement.
- Participant B considered cost as constraint as medium for product requirement, organization requirement and external requirement.

Conclusion

Cost influences in selection of software architectural decision. The rationale for consideration of cost is due to determination of licenses cost of implementation

technology and deployment environment including cost of software development team therefore requirement of making budgets.

4.2.10 Stakeholders as Assumption or Constraint

4.2.10.1 Stakeholders as Consideration

Influence on Selection of Architecture Style

- Participant A considered stakeholders consideration as medium for product requirement, organization requirement and external requirement.
- Participant B considered stakeholders consideration as low for product requirement, medium for organization requirement and high external requirement.

Influence on Selection of Tactics

- Participant A considered stakeholders consideration as medium for product requirement, high for organization requirement and medium for external requirement.
- Participant B considered stakeholders consideration as high for product requirement, low for organization requirement and high for external requirement.

Influence on Reference Architecture

- Participant A considered stakeholder consideration as medium for product requirement, organization requirement and external requirement.
- Participant B considered stakeholder consideration as high for product requirement, organization requirement and external requirement.

4.2.10.2 Stakeholders as Constraint

Influence on Selection of Architecture Style

- Participant A considered stakeholders as constraint as medium for product requirement, organization requirement and external requirement.
- Participant B considered stakeholders as constraint as high for product requirement and organization requirement and medium for external requirement.

Influence on Selection of Tactics

- Participant A considered stakeholders as constraint as medium for product requirement, and high for organization requirement and external requirement.

- Participant B considered stakeholders as constraint as high for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered stakeholders as constraint as high for product requirements, low for organization requirements and high external requirement.
- Participant B considered stakeholders as constraint as high for product requirements, organization requirements and external requirements.

4.2.10.3 Stakeholders as Assumption

Influence on Selection of Architecture Style

- Participant A considered stakeholders as assumption as mediumfor product requirements, organization requirements and external requirement.
- Participant B considered stakeholders as assumption as medium for product requirements, organization requirements and external requirement.

Influence on Selection of Tactics

- Participant A considered stakeholders as assumption as low for product requirements, medium for organization requirements and high for external requirement.
- Participant B considered stakeholders consideration as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered stakeholders as assumption as mediumfor product requirements and low for organization requirements and external requirement.
- Participant B considered stakeholders as assumption as medium for product requirements, organization requirements and external requirement.

Conclusion

Stakeholders influences in selection of software architectural decision. The rationale for consideration of stakeholders is due to interest or investment in certain architectures due to existing systems, or reusing existing hardware infrastructure or already purchased extra licenses of certain implementation technology or deployment environment.

4.2.11 Deployment topologies as Assumption or Constraint

4.2.11.1 Deployment topologies as Consideration

Influence on Selection of Architecture Style

- Participant A considered deployment topologies consideration as high for product requirement, medium for organization requirement and external requirement.
- Participant B considered deployment topologies consideration as high for product requirement, medium for organization requirement and high for external requirement.

Influence on Selection of Tactics

- Participant A considered deployment topologies consideration as high for product requirement and organization requirement and low for external requirement.
- Participant B considered deployment topologies consideration as medium for product requirement and organization requirement and low for external requirement.

Influence on Reference Architecture

- Participant A considered deployment topologies consideration as medium for product requirement, high for organization requirement and mediumfor external requirement.
- Participant B considered deployment topologies consideration as high for product requirement and medium for organization requirement and external requirement.

4.2.11.2 Deployment topologies as Constraint

Influence on Selection of Architecture Style

- Participant A considered deployment topologies as constraint as medium for product requirement, organization requirement and external requirement.
- Participant B considered deployment topologies as constraint as high for product requirement and medium for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered deployment topologies as constraint as high for product requirement and organization requirement and medium for external requirement.
- Participant B considered deployment topologies as constraint as high for product requirement and medium for organization requirement andexternal requirement.

Influence on Reference Architecture

- Participant A considered deployment topologies as constraint as high for product requirement and medium for organization requirement and external requirement.
- Participant B considered deployment topologies as constraint as high for product requirement and organization requirement and medium for external requirement.

4.2.11.3 Deployment topologies as Assumption

Influence on Selection of Architecture Style

- Participant A considered deployment topologies as assumption as medium for product requirements and low for organization requirements and external requirement.
- Participant B considered deployment topologies consideration as medium for product requirement, low for organization requirement and medium for external requirement.

Influence on Selection of Tactics

- Participant A considered deployment topologies as assumption as high for product requirements and low for organization requirements and external requirement.
- Participant B considered deployment topologies consideration as medium for product requirement, organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered deployment topologies as assumption as low for product requirements, organization requirements and external requirement.
- Participant B considered deployment topologies as assumption as medium for product requirements low for organization requirements and medium for external requirement.

Conclusion

Deployment topologiesinfluences in selection of software architectural decision. The participants declared the rationales for this factor is same as that of deployment environment.

4.2.12 Application domain as Assumption or Constraint

4.2.12.1 Application Domain as Consideration

Influence on Selection of Architecture Style

- Participant A considered application domain consideration as high for product requirement organization requirement and external requirement.
- Participant B considered application domain consideration as medium for product requirement and high for organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered application domain consideration as high for product requirement and organization requirement and medium for external requirement.
- Participant B considered application domain consideration as high for product requirement and organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered application domain consideration as medium for product requirement and high for organization requirement and external requirement.
- Participant B considered application domain consideration as high for product requirement, organization requirement and external requirement.

4.2.12.2 Application domain as Constraint

Influence on Selection of Architecture Style

- Participant A considered application domain as constraint as high for product requirement medium for organization requirement and high for external requirement.
- Participant B considered application domain as constraint as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered application domain as constraint as high for product requirement medium for organization requirement and high for external requirement.
- Participant B considered application domain as constraint as high for product requirement medium for organization requirement and high for external requirement.

Influence on Reference Architecture

- Participant A considered application domain as constraint as high for product requirements, organization requirements, and external requirement.
- Participant A considered application domain as constraint as medium for product requirements, organization requirements, and external requirement.

4.2.12.3 Application Domain as Assumption

Influence on Selection of Architecture Style

- Participant A considered application domain as assumption as medium for product requirements and low for organization requirements and external requirement.
- Participant B considered application domain consideration as medium for product requirement, organization requirement and external requirement.

Influence on Selection of Tactics

- Participant A considered application domain as assumption as high for product requirements, and low for organization requirements and external requirement.
- Participant B considered application domain consideration as low for product requirement and medium for organization requirement and external requirement.

Influence on Reference Architecture

- Participant A considered application domain as assumption as low for product requirements, organization requirements and external requirement.
- Participant B considered application domain as assumption as medium for product requirements, organization requirements and external requirement.

Conclusion

Application domain influences in selection of software architectural decision. The rationale for consideration is due to criticality of certain domains like financial, e-commerce based web site or certain real timesystems have certain mission critical needs.The decision whether to go with certain reference architecture or not is solely dependent on application domain.

4.3 Design Decisions

This section describes the design decisions taken during software architecture design process.

Design Decision # 1

Reference architecture will be not applied related to the domain of application. A thorough assessment was conducted in order to find the best suitable reference architecture. A reference architecture which is the closest match was considered. A brainstorming session was conducted in order to decide whether to go with the reference architecture or not. After series of examination it was decided as reference architecture will not be applied. The rationale for not applying reference architecture is due to:

1. High cost of implementation.
2. It is difficult to achieve iterations plan within defined time which is agreed upon by all the stakeholders.
3. The reference architecture requires certain components that need to develop with a particular implementation technology, which is beyond the skill set of project development teams available within organization.
4. As the system will be comprised of various application types like windows, web and mobile so architects based on their analysis finalized that it requires an agent component on top of reference architecture, which results addition of another layer on reference software architecture.

Design Decision # 2

There is a need that different components of different application type of the system communicate with each other in a real time fashion with certain defined business goals. **Message Bus Architecture style was decided to be used** in order to receive and send messages using one or more communication channels, so that application of different application type can interact without having to know specific details about each other. The rationales for using this decision are:

1. The stakeholders require a flexible solution that is capable of adding and removing features as components in their final product.

2. The communication channels are required to consider different transport protocols like TCP/IP and UDP as per business rules of same application types i.e. windows application.
3. By using a message-based communication mode the resulting system will interact with applications types as well as domains developed for different deployment environments, using different implementation technologies like Microsoft .NET and Java.

Design Decision # 3

The software application produces and consumes data. This data is of two types one a temporary data used for communication among components and secondly a persistent data. In addition the need for the availability of persistent data is of high degree importance. So to achieve this goal **client-server architecture style was used, for interaction between data repository and components requiring data.**

1. A database server was used to serve data based on demands. The reason for introducing proper database management system as a server in this case is due to internal mechanisms of concurrency management of database system. Also data requirement originates from various different application types.
2. The reason for introducing client server architecture style is also due to savetime as in case of using flat files as a data repository requires additional programming to manage multithreading. In addition multithreading programming requires more experienced programmers in software development team.
3. The application requires centralize data storage, backup for effective management functions.

Design Decision # 4

Service oriented architecture style was used for such communication of components.

1. As the tiers hosting components have different deployment environment like operating systems.
2. The development team has expertise in implementation technology by using which they can achieve use cases in less time thus saving cost.

Design Decision # 5

There is a need of high reliability and integrity of data which is served by data server. The application components communicating with the server requires complete acknowledgements of their successful transactions. So TCP/IP communication protocol was used in order to achieve reliability consideration. In addition there are certain components that are communicating but require quick response. Therefore UDP protocol was used in that case. The rationales for the decisions are as follows:

1. The application components hosted on mobile devices have windows mobile edition which offers very limited support for hosting complex application capable of managing queue, therefore to achieve fast communication in order to achieve performance goal UDP is used.
2. Certain components will be operating wireless environment.

Design Decision # 6

There is a need of back up and monitoring system by various stakeholders. This system is declared as a sub system which of application type of LAN based windows application. The sub stem is also responsible in case of disaster recovery. The architecture style for this sub system was finalized as **N-Tier Architecture**. The rationale for this decision is:

> Application type for this component was windows based data driven application. This component incorporated certain business rules required for deployment of application to new sites with respect to different application types and deployment environments.

Design Decision # 7

There is a need of high availability and to achieve this goal **ping/echo tactics** is used for the components operating under wired network.The rationale for this decision is:

> Considering criticality of application domain the tactics is implemented as software development team have already experience of implementation of ping/echo tactics by using required implementation technology targeting desired deployment environment.

Design Decision # 8

There isneed of high availability and to achieve this goal **heartbeat tactics** is used for components operating under wireless network on mobile devices. The rationale for this decision is:

> Considering criticality of application domain the tactics is implemented as the target component will be residing on mobile device which has windows mobile operating system. In addition, deployment topologies areadhoc wireless network. Therefore deployment environment and topology are the basis for using heartbeat tactics.

Design Decision # 9

There is a need of high performance and to achieve this goal **increase computational efficiency tactics**was used. The rationale for this decision is:

> Considering criticality of application domain the tactics is implemented so as to develop a product that is capable of high and fast performance on machines with low specifications, therefore reducing the cost of deployments.

Design Decision # 10

There is a need of fastresponse time and availability to achieve this goal **concurrency tactics** was used. The rationale for this decision is:

> Considering criticality of application domain the tactics is implemented so as to fully utilize the available computational resources smartly as per wish of an influential stakeholder. This also eliminates the excessive use of queuing mechanisms which required extra development time and system software from specific vendors.

Design Decision # 11

There is a security need to achieve goals of resistance of attacks. Therefore **authenticate tactics** is used. The rationale for this decision is:

> Considering criticality of application domain the tactics is implemented so as to meet the future need of biometric identification of users which is currently achieved by passwords.

Design Decision # 12

There is a security need to achieve goals providing limited access to features of application. Therefore **authorize tactics** is used. The rationale for this decision is:

> Considering criticality of application domain the tactics is implemented so as meet the requirement in less amount of time as software development team has already vast experience of incorporating this tactics to various application types.

Design Decision # 13

There is a requirement from software development organization that software must be maintainable and scalable.**Object oriented architecture style**was considered to meet this goal. The rationale for this decision is:

1. Software development team is using implementation technologies which are based on object oriented principles. In addition, certain APIs need to be used by software development team which all are developed using object oriented principles and can be reused only with implementation technology supporting object orientation.
2. Stakeholders envisioned a system dynamic enough to operate with or without certain components. In addition, theversions of different components must operate.

Note:

Organization processes and global software development factors regarding the case are known to participants at the time of decision and both participants have the agreement that these two factors influence software architectural decisions.

5 Model

This research answers the following question:

> What contextual knowledge software architects require for reusability of technical knowledge of software architecture?

Case study was conducted for identification of contextual knowledge elements. A thorough analysis on case study results and design decisions was performed to identify the relationships of contextual knowledge elements with other constructs of software architecture knowledge elements. On the basis of analysis of results and design decisions following model has been proposed that improves the reusability of software architecture knowledge.

5.1 A Model for Reusability of Software Architectural Knowledge

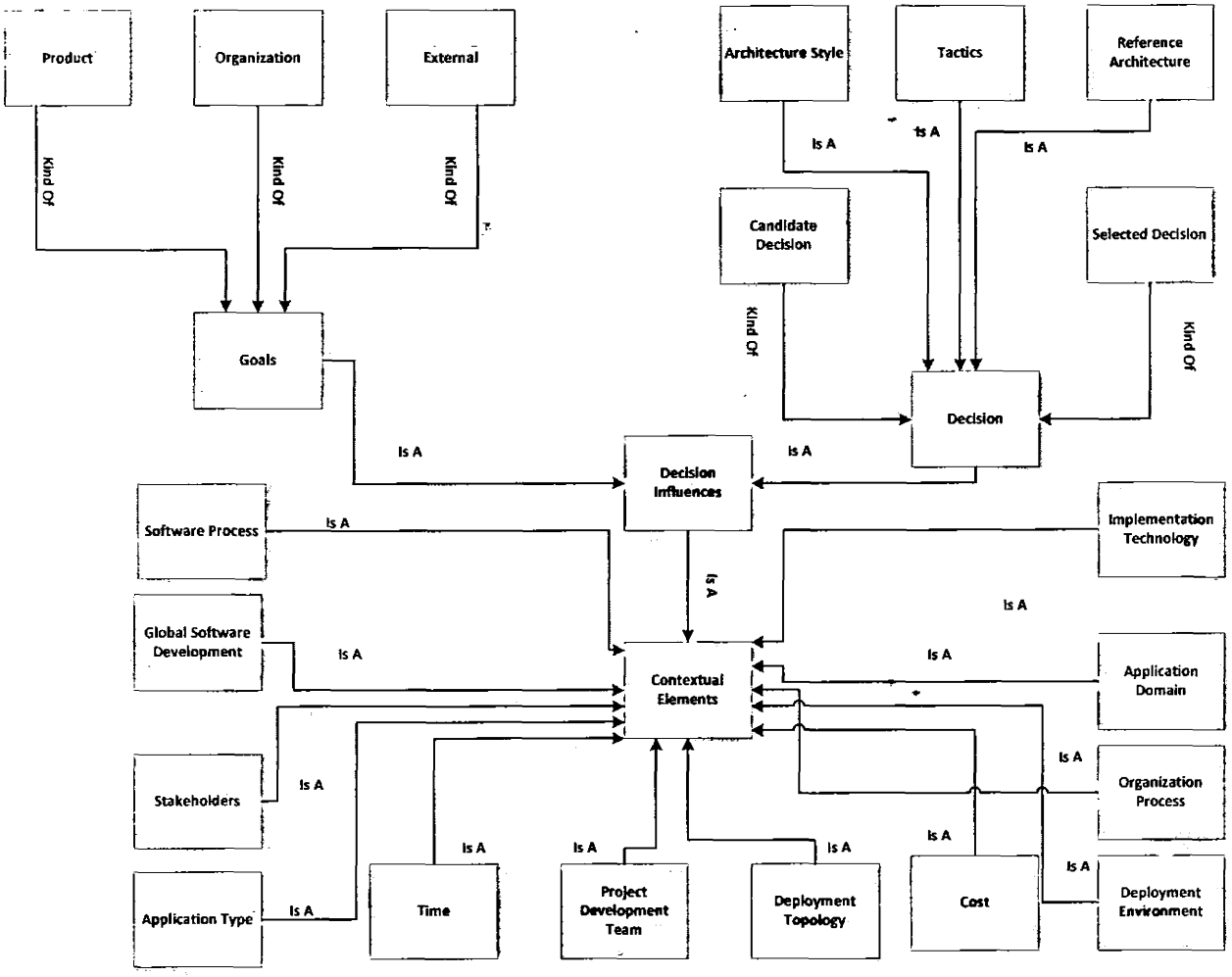


Figure 5.1 A Model for Reusability of Software ArchitecturalKnowledge

5.1.1 Description of Proposed Model

The goal describes goals of applicationunderconsideration and three types of goals are product, organization and external.Contextual elements describefactors that determine selection of software architecture knowledge. The constituents of knowledge elements are cost, implementation technology, application domain, organization process, deployment environment, deployment topology, project development team, time, application type, stakeholders, global software development and software process. These contextual knowledge elements are either of two type assumptions or constraints. The decision influences determines selection of decision which includes architecture style, reference architecture or tactics. In order to enhance reusability the decisions are categorized as candidate decision or selected decision.The model is represented by semantic network [42] way of knowledge representation.

5.1.2 Characteristics of Proposed Model

The proposed model helps improve reusability of software architectural knowledge.The model associates contextual factors with the software architecture decision. This association incorporates influences of decisions in form of structured contextual knowledge elements. In literature all the available models of software architecture knowledge management offers limited reusability as all models lacks association of decisions with contextual knowledge elements. The focus of this model is effective management of knowledge in such a way that helps architects in making new more informed decisions. This helps in organizational learning asavailability of such structured knowledge empowers even less experienced architects to make correct decisions.

5.1.3 Limitation

Although the proposed model serves the basic needs of reusability, this model have the following limitations:

1. Degree of influence on decision of all the contextual factors needs to be determined. The determination of degree of influences improves the associations.

2. As software architecture design process is an iterative process, so this model currently stores only final decisions and does not associates decision with respect to different iterations.
3. The proposed model currently lacks complete and final version of tool support.

5.2 Proof of Concept

Based on the model a prototype proof of concept was developed in order to validate the implementation of model. In this screen contextual knowledge elements are captured against each technical knowledge element. In addition, after selection of contextual element and possible value proof of concept also generates suggestions. The suggestions help the architect to apply the best available solution to meet the requirement.

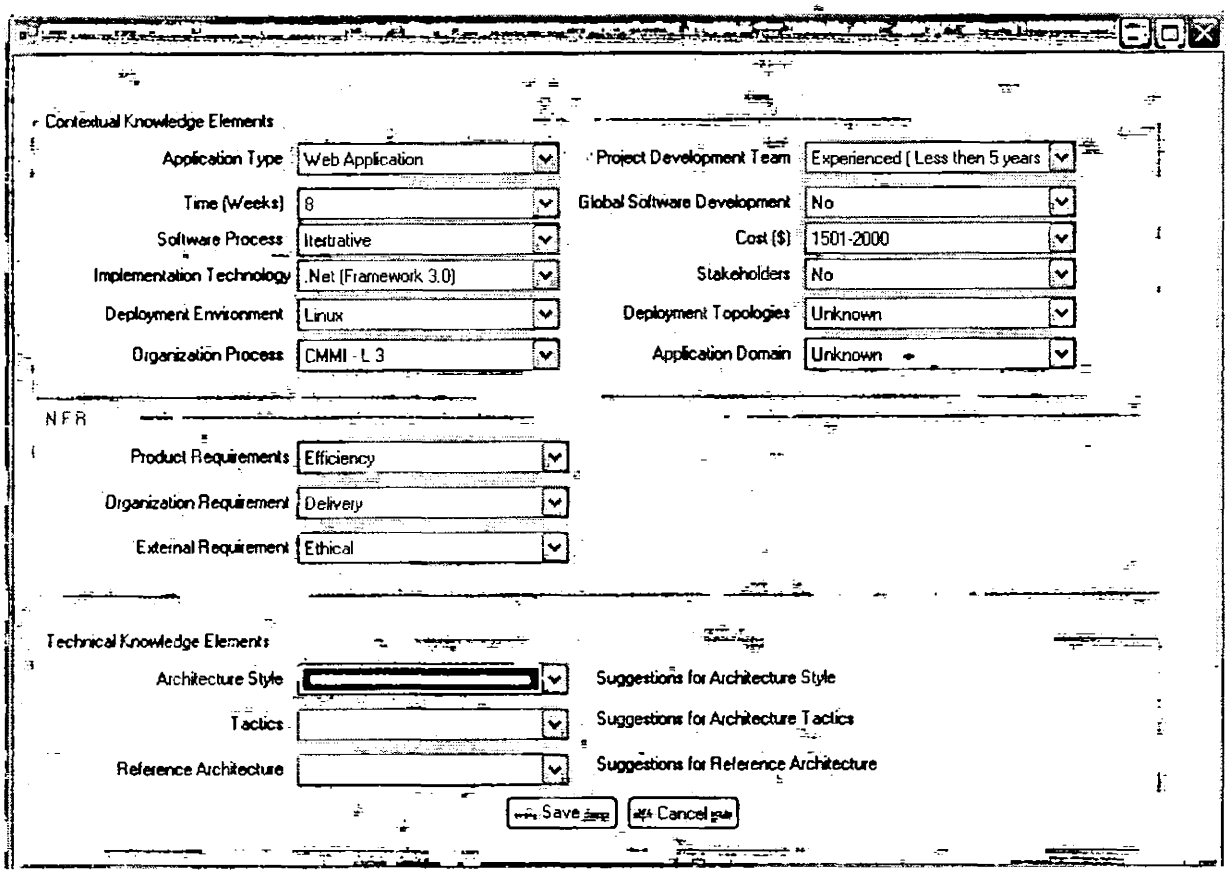


Figure 5.2 Screen of POC that stores decisions and elaboration of reusability

6 Conclusion and Future Work

In this section, the summary of this research has been explained alongwith the thesis contributions. Moreover, the research questions have also been answered. Enhancements that can be done in this work are also suggested.

6.1 Summary

Software architecture design is knowledge intensive process that produces and requires knowledge. Commonly during the architecture development process, decisions are not documented explicitly but are reflected by the models the architects build, consequently, useful knowledge attached to the decisions and its process is lost. The software architecture knowledge can be categorized as technical knowledge [2] (such as patterns, tactics, and quality attribute analysis models) and contextual knowledge (design rationale) [2].

Software architecture embodies significant decisions, these decisions are in the form of tacit knowledge, but rationales behind the decisions are not available. This causes two main problems:design decisions vaporize;reusability of technical knowledge applied in designing similar software architecture is difficult.

This research work has answered followingthe following question:

1. What contextual knowledge software architects require for reusability of technical knowledge of software architecture?
2. How to preserve contextual knowledge for reusability of software architecture technical knowledge?

6.2 Contribution

This research is intended to identify the contextual knowledge elements used by software architects during software architecture design. This research also included influences of assumptions and constraints of these identified contextual knowledge elements in selection of technical knowledge elements i.e. architecture style, tactics or reference architecture. The contextual knowledge elements are identified on the basis of analysis performed on case study results. The proposed model depicts arrangements of these software architecture knowledge elements. Hence decision and rationales are codified which results in reusability of technical

knowledge elements in other related scenarios. The research identified the reusability needs of software architects.

Although the proposed model serves the basic needs of reusability, this model have the following limitations, degree of influence on decision of all the contextual factors needs to be determined.As software architecture design process is an iterative process, so this model currently stores only final decisions and does not associates decision with respect to different iterations. The proposed model currently lacks complete and final version of tool support.

6.3 Future work

Future work for this research is to study:

- The degreeon which each identified contextual knowledge element influences in selection of technical knowledge element.
- To study any type of non functional requirement and to determine what type of non functional requirements are given more importance during trade off analysis.
- To develop a full version tool for software architecture knowledge management based on the proposed model.

7 References

1. Hans van Vliet.Software Architecture Knowledge Management, 19 Australian Conference on Software Engineering. 2008

2. Muhammad Ali Babar . The Application of Knowledge-Sharing Workspace Paradigm for Software Architecture Process. SHARK 08, 2008

3. Muhammad Ali Babar, Ian Gorton and Ross Jeffery. Capturing and Using Software Architecture for Architecture Based Software Development. Proceedings of the Fifth International Conference on Quality Software (QSIC'05), 2005

4. Anton Jansen, Jan Bosch. Software Architecture as a Set of Architectural Design Decisions, Proceedings of the 5th IEEE/IFIP Working Conference on Software Architecture (WICSA 2005), pp. 109-119, November 2005.

5. Patricia Lago, Paris Avgeriou, Rafael Capilla, Philippe Kruchten, Wishes and Boundaries for a Software Architecture Knowledge Community, Seventh Working IEEE/IFIP Conference on Software Architecture 2008

6. Muhammad Ali Babar, Ian Gorton Architecture Knowledge Management: Challenges, Approaches, and Tools, 29th International Conference on Software Engineering (ICSE'07 Companion), 2007

7. Remco C. de Boer, RikFarenhorst In Search of Architectural Knowledge. SHARK 08, 2008

8. Ibrahim Habli, Tim Kelly Capturing and Replaying Architectural Knowledge through Derivational Analogy SHARK 07, 2007

9. David Falessi, Martim Becker, Giovanni Cantone, Design Decision Rationale: Experiences and Steps Ahead Towards Systematic Use. SHARK 06, 2006

10. Patricia Lago, Paris AvgeriouFirst Workshop on Sharing and Reusing Architectural Knowledge. SHARK 06, 2006

11. Muhammad Ali Babar, Remco C. de Boer, TorgeirDingsøyr, RikFarenhorst Architectural Knowledge Management Strategies: Approaches in Research and Industry. SHARK 07. 2007

12. M. Ali Babar, I. Gorton, and R. Jeffery. Toward a Framework for Capturing and Using Architecture Design Knowledge. Technical report, The University of New South Wales, June 2005.

13. Len Bass, Paul Clements, Rick Kazman , Software Architecture in Practice, Second Edition

14. H. Choi, Y. Choi, and K. Yeom, An Integrated Approach to Quality Achievement with Architectural Design Decisions, Journal of Software, vol. Volume 1, pp. 40-49, 2006.

15. T. Al-Naeem, I. Gorton, M. A. Babar, F. Rabhi, and B. Benatallah, A quality-driven systematic approach for architecting distributed software applications, in Software Engineering, 2005. ICSE 2005. Proceedings. 27thInternational Conference on, 2005, pp. 244-253. 2005

16. Aman-ul-haq, Muhammad Ali Babar, Tool Support for Automating Architectural Knowledge Extraction SHARK May 2009

17. Anton G. J. Jansen, Jan van der Ven, Paris Avgeriou, Dieter K. Hammer, Tool support for Architectural Decisions, Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA 2007), January 2007.

18. M. Sinnema, J. S. van der Ven, S. Deelstra, Using Variability Modeling Principles to Capture Architectural Knowledge, Proceedings of the Workshop on Sharing and Reusing architectural Knowledge (SHARK2006), June 2006 .

19. Remco C. de Boer, Rik Farenhorst1, Patricia Lago, Hans van Vliet, Viktor Clerc, and Anton Jansen, Architectural Knowledge: Getting to the Core, QoSA 2007, LNCS 4880, pp. 197–214, 2007.Springer-Verlag Berlin Heidelberg 2007.

20. Muhammad Ali Babar, Ian Gorton A Tool for Managing Software Architecture Knowledge. 29th International Conference on Software Engineering Workshops(ICSEW'07)

21. R. Capilla, F. Nava, S. P´erez, and J. C. Due˜nas. A Web-based Tool for Managing Architectural Design Decisions. In 1st ACM Workshop on Sharing Architectural Knowledge (SHARK), Torino, Italy, 2006.

22. http://www.archium.net

23. Per Runesn, Martin Host, Guidelines for conducting and reporting case study research in software engineering. This article is published with open access at springerlink.com, 2008

24. Bjorn Regnell, Martin Host, Johan Nattoch Dag, An industrial Case Study on Distributed Prioritization in Market Driven Requirements Enfineering for Packeged Software. Requirement Eng (2001) Springer-Verlag London. 2001

25. Software engineering institute software architecture definition page: http://www.sei.cmu.edu/architecture/definitions.html.

26. I. Sommerville. Software Engineering. Addision-Wesley, 8 edition, 2007.

27. Murray Jennex. Case Studies in Knowledge Management.

28. D. Falessi, G. Cantone, and M. Becker. Documenting design decision rationale to improve individual and team design decision making: an experimental evaluation.ACM/IEEE international symposium on International symposium on empirical software engineering (ISESE '06), pages 134–143, New York, NY, USA, 2006. ACM Press. 2006

29. P. Kruchten, P. Lago, and H. van Vliet. Building up and reasoning about architectural knowledge. In Proceedings of the Second International Conference on the Quality of Software Architectures (QoSA 2006), 2006.

30. Jason Baragry and Karl Reed. Why Is It So Hard To Define Software Architecture? School of Computer Science and Computer Engineering La Trobe University Bundoora, Vic 3083, Australia

31. Yujian Fu, Zhijiang Dong and Xudong He. An Approach to Validation of Software Architecture Model. Proceedings of the 12th Asia-Pacific ,Software Engineering Conference (APSEC'05), 2005

32. Microsoft Application Architecture Guide 2.0 Patterns and Practices.

33. Winston Tellis ,Introduction to Case Study, The Qualitative Report, Volume 3, Number 2, July, 1997

34. Charles L. Chen, Danhua Shao, Dewayne E. Perry. An Exploratory Case Study Using CBSP and Archium. Proceedings of the Workshop on Sharing and Reusing architectural Knowledge (SHARK2006), 2006.

35. William J. Koscho, William Ries. Identifying and Proactively Managing Architecture Risk. LMSA'09, Vancouver, Canada. 2009

36. Nary Subramanian, Lawrence Chung "Relationship between the Whole of Software Architecture and its Parts:An NFR Perspective"Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN), 2005

37. Yujian Fu, Zhijiang Dong A Method for Realizing Software Architecture Design, Proceedings of the Sixth International Conference on Quality Software (QSIC'06) 2006

38. JoostNoppen , DalilaTamzalit, ETAK: Tailoring Architectural Evolution by (re-)using Architectural Knowledge, SHARK'10 May 2, Cape Town, South Africa, 2010

39. Walter F. Tichy, Should Computer Scientists Experiment More? IEEE 1998.

40. Dooley, K. "Simulation Research Methods," Companion to Organizations, Joel Baum (ed.), 2002, London: Blackwell, p. 829-848.

41. http://www.wordiq.com/definition/Survey_research

42. Stephan Grimm, Pascal Hitzler, Andreas Abecker, Knowledge Representation and Ontologies Logic, Ontologies and SemanticWeb Languages.

43. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.8581

8 Glossary

AK - Architectural Knowledge

KM – Knowledge Management

NFR-Non-Functional Requirements

CMMI-Capability Maturity Model Integration

Appendix A

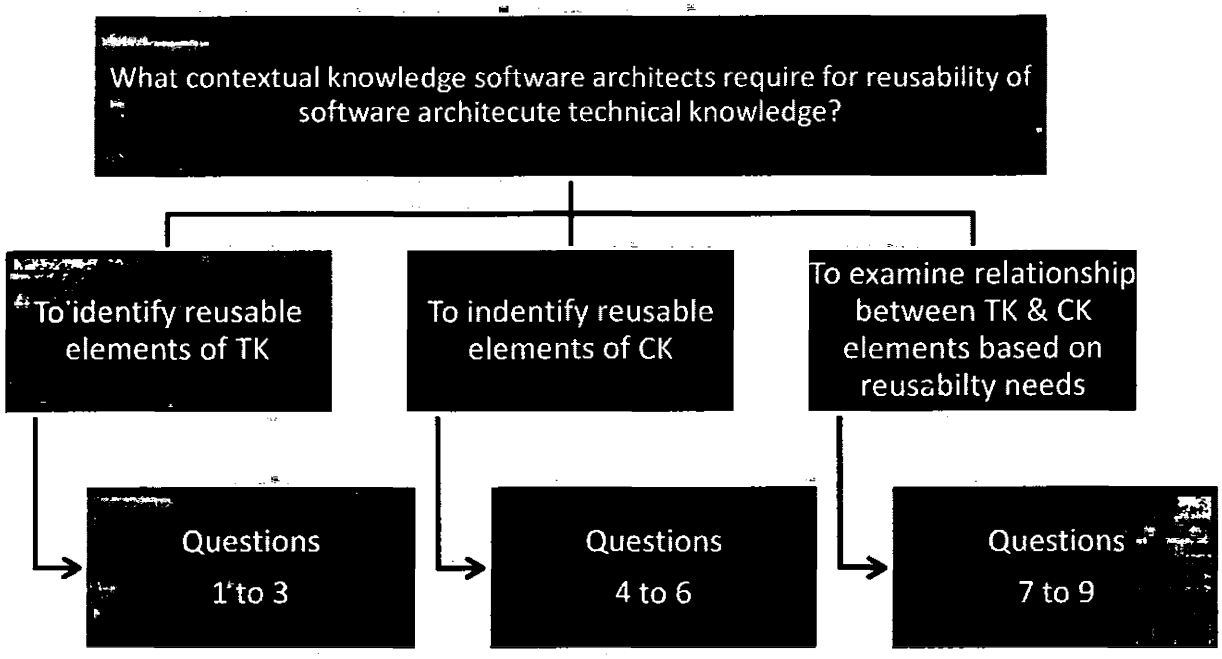
Questionnaire

Organization of Questions on the Basis of Goals & Sub-Goals

This reseach work is to answer following questions:

1. **What contextual knowledge software architects require for reusability of technical knowledge of software architecture?**
2. **How to preserve contextual knowledge for reusability of software architecture technical knowledge?**

In order to answer first question, a case study was executed and a questioner is formed with following goal and sub goals:



Questions

Identification of Technical Knowledge Elements						
No	Questions	Answer Options				
		Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
1	Is selection of particular tactics involves in architecture design?					
2	Is selection of particular architecture style occurs in architecture design?					
3	Is consideration of reference architecture occurs in architecture design?					

4. Up to what degree following factors influences selection of particular software architecture knowledge element to satisfy particular non-functional requirement of type product requirement (efficiency, reliability, portability, usability, performance etc)

No	Factor	Tactics			Architecture Style			Reference Architecture		
		Low	Medium	High	Low	Medium	High	Low	Medium	High
1	Application Type									
2	Time									
3	Software Process									
4	Implementation Technology									
5	Deployment Environment									
6	Organization Processes									
7	Project Development Team									
8	Global Software Development									
9	Cost									
10	Stakeholders									
11	Deployment Topologies									
12	Application Domain									

5. Up to what degree following factors influences selection of particular software architecture knowledge element to satisfy particular non-functional requirement of type organizational requirement (delivery, implementation, standards etc)

No	Factor	Tactics			Architecture Style			Reference Architecture		
		Low	Medium	High	Low	Medium	High	Low	Medium	High
1	Application Type									
2	Time									
3	Software Process									
4	Implementation Technology									
5	Deployment Environment									
6	Organization Processes									
7	Project Development Team									
8	Global Software Development									
9	Cost									
10	Stakeholders									
11	Deployment Topologies									
12	Application Domain									

6. Up to what degree following factors influences selection of particular software architecture knowledge element to satisfy particular non-functional requirement of type external requirement (interoperability, ethical, legislative, safety etc)

No	Factor	Tactics			Architecture Style			Reference Architecture		
		Low	Medium	High	Low	Medium	High	Low	Medium	High
1	Application Type									
2	Time									
3	Software Process									
4	Implementation Technology									
5	Deployment Environment									
6	Organization Processes									
7	Project Development Team									
8	Global Software Development									
9	Cost									
10	Stakeholders									
11	Deployment Topologies									
12	Application Domain									

7-Mark as H for high, M for medium or L for low against the following factors when considered as an assumption or constraint in selection of particular knowledge element to satisfy particular non-functional requirement of type product requirement (efficiency, reliability, portability, usability, performance etc)

No	Factor	Tactics		Architecture Style		Reference Architecture	
		Assumption	Constraint	Assumption	Constraint	Assumption	Constraint
1	Application Type						
2	Time						
3	Software Process						
4	Implementation Technology						
5	Deployment Environment						
6	Organization Processes						
7	Project Development Team						
8	Global Software Development						
9	Cost						
10	Stakeholders						
11	Deployment Topologies						
12	Application Domain						

8-Mark as H for high, M for medium or L for low against the following factors when considered as an assumption or constraint in selection of particular knowledge element to satisfy particular non-functional requirement of type organizational requirement (delivery, implementation, standards etc)

No	Factor	Tactics		Architecture Style		Reference Architecture	
		Assumption	Constraint	Assumption	Constraint	Assumption	Constraint
1	Application Type						
2	Time						
3	Software Process						
4	Implementation Technology						
5	Deployment Environment						
6	Organization Processes						
7	Project Development Team						
8	Global Software Development						
9	Cost						
10	Stakeholders						
11	Deployment Topologies						
12	Application Domain						

9-Mark as H for high, M for medium or L for low against the following factors when considered as an assumption or constraint in selection of particular knowledge element to satisfy particular non-functional requirement of type external requirement (interoperability, ethical, legislative, safety etc)

No	Factor	Tactics		Architecture Style		Reference Architecture	
		Assumption	Constraint	Assumption	Constraint	Assumption	Constraint
1	Application Type						
2	Time						
3	Software Process						
4	Implementation Technology						
5	Deployment Environment						
6	Organization Processes						
7	Project Development Team						
8	Global Software Development						
9	Cost						
10	Stakeholders						
11	Deployment Topologies						
12	Application Domain						