A Comparative Analysis of Architecture Frameworks

Antony Tang   Jun Han
School of Information Technology
Swinburne University of Technology
Melbourne, Australia
Email: {atang,jhan}@it.swin.edu.au

Pin Chen
DSTO C3 Research Centre
Department of Defence
Canberra, Australia
E-mail: Pin.Chen@dsto.defence.gov.au

Abstract

Architecture frameworks are methods used in architecture modeling. They provide a structured and systematic approach to designing systems. To date there has been little analysis on their roles in system and software engineering and if they are satisfactory. This study provides a model of understanding through analyzing the goals, inputs and outcomes of six Architecture Frameworks. It characterizes two classes of architecture frameworks and identifies some of their deficiencies. To overcome these deficiencies, we propose to use costs, benefits and risks for architecture analysis. We also propose a method to delineate architecture activities from detailed design activities.

1. Introduction

This paper investigates the concept of architecture by examining six Architecture Frameworks (AF). Architecture plays a major role in the development of information systems. The act of architecture design in the development cycle is generally understood to be systematic analysis and design of related information to provide a model for guiding the actual development of information systems.

Researchers have offered various definitions and explanations of architecture. Garlan and Shaw [1] suggested that software architecture is concerned with issues beyond algorithms and data structures of computation. Perry and Wolf [2] distinguished architecture from design by suggesting that architecture is concerned with the selection of architectural elements, their interaction and their constraints, but design is concerned with the modularization and detailed interfaces of the design element. Monroe et al. [3] suggested that architecture is not about details of implementation. IEEE's definition of architecture states that it is "the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution" [4]. There was also an attempt to formally distinguish architecture activities from design activities [5].

Architecture frameworks have helped to improve the understanding of the subject by providing systematic approaches to architecture development, but many aspects of architecture remain ambiguous. The ambiguities are the following. (a) The scope of architecture - should the scope of architecture encompass software components only or include other aspects of information system development? (b) The role of an architect - architect's role in the system development life cycle is often unclear. Architect could take on roles of a business analyst, a software designer or a system analyst. (c) The outcomes – what should be the outcome of architecture activities? Outcomes may range from business functions documents to detailed program designs. (d) Architecture activities – architecture activities involve design and modeling, but what level of detail belongs to architecture and when do detailed design activities start? (e) Verifying architecture - to what extent should and could outcomes of architecture be measured, verified or validated? (f) When to engage - system of what size and complexity would require architecture? Do systems of various sizes and complexities require same architecture outcomes? (g) Level of architecture – what is the relationship between enterprise architecture and stand-alone system architecture?

The contributions of this paper are the systematic analysis and comparison of AFs using generic architecture characteristics or elements. As such, a model of understanding for architecture is described to clarify some of the ambiguity. Analysis of architecture frameworks has identified some deficiencies and we

propose several ways to overcome them. Section 2 introduces the goals of architecture framework. Section 3 defines a neutral way to study the frameworks. Section 4 presents the study of six architecture frameworks. Section 5 analyses and compares frameworks, discusses their deficiencies and proposes solutions. We conclude our findings in section 6.

2. Architecture Frameworks

Architecture modeling commonly uses high level abstraction called views. AFs use viewpoints to create views that represent different perspectives of a system model. Common viewpoints are business architecture, information architecture, software architecture and technical architecture. Specific frameworks being studied may have underlying goals to focus on distributed systems, enterprise architecture or industry specific systems. After examining different architecture frameworks and the IEEE Recommended Practice for Architectural Description of Software-Intensive System [4], this paper puts forward a set of common goals for AF. These goals are independent of industry domain, architecture style and system size.

- Architecture Definition and Understanding – make use of standard terms, principles and guidelines for consistent application of the framework for the communication of architecture information to stakeholders.
- Architecture Process – employ a well-defined process to guide the construction of architecture
- Architecture Evolution Support – employ processes and mechanisms that support systems evolution
- Architecture Analysis – provide a set of viewpoints to guide the collection and analysis of information for making architecture choices
- Architecture Models – provide consistent standards to document architecture specifications for the planning, management, communication and execution of activities related to system development
- Design Tradeoffs – select a design from more than one design choices by resolving multi-dimensional conflicting requirements
- Design Rationale – document reasons behind design decisions for verification, i.e. "architect for a reason" [6]
- Standardization – ensure development and architectural standards are maintained

- Architecture Knowledge Base – provide consistent representation and repository of design and architecture design rationale
- Architecture Verifiability – provide sufficient information or explanation in the architecture design for review and verification

These common goals collectively provide an objective guideline for the selection and use of AF. As such, resulting architecture models produced by AF would provide consistent representation of architecture models.

There are a number of established AFs. The Zachman Framework for Enterprise Architecture [7] is one of the earliest works in this area. *4+1 View Model of Architecture* is a framework for modeling software architecture [8]. The US Federal Government issued a standard Federal Enterprise Architecture Framework version 1.1 [9]. The International Standards Organization (ISO) in conjunction with the International Telecommunication Union (ITU-T) issued a Reference Model for Open Distributed Computing (RM-ODP) [10]. The Open Group, an industry standard organization, issued The Open Group Architectural Framework (TOGAF) version 8.1 in 2003 [11]. The US Department of Defense released the DoD Architecture Framework version 1.0 [12] for all architectures developed within the DoD to comply with. The selection of AF in this study is based on frameworks that are well known in the relevant communities. The intention of this study is not to report details of each AF but to analyze and compare their similarities and differences.

3. Basis for Analysis

To architect a complex system involves considerations of multiple dimensions such as business requirements, technical requirements, costs, current architecture and future architecture etc. [13]. The dimensions, or inputs, to the architecture process are themselves interrelated and cannot be considered in isolation in the process. For instance, architecting a system with flexibility and portability may have an impact on performance, cost and schedule. A key outcome of architecture is to create a model of architecture designs. The model considers complex dimensions so that balanced tradeoffs are reached and the risks of achieving the system's objectives are minimized. Risks that arise from construction or evolution of a system may lie in many areas. They represent the uncertainty of achieving the system objectives. Architecture activities remove major uncertainties through modeling and specification of the

system to the point where the problem to be solved become well understood and the modeled solution has a high certainty of achieving its objectives.

In this paper, we propose to analyze AF in terms of their goals, inputs and outcomes. *Goals* are described in the previous section. *Inputs* represent information that architecture modeling considers. *Outcomes* represent results and deliverables. Typical inputs to architecture activities are the following.

- Business Drivers – business goals, direction, principles, strategies and priorities
- Technology Inputs – strategic architecture direction including technology platforms, future architecture, systems interoperability and emerging technology standards
- Business Requirements – users' requirements, functional requirements, data requirements and other business system related requirements
- Information System Environment – budget, schedule, technical constraints, resources and expertise, organization structure, other constraints, enterprise knowledge base
- Current Architecture – current standards and infrastructure
- Non Functional Requirements – some of these requirements are also referred to as Quality Attributes (QA) or Quality of Services (QoS). These requirements include availability, reliability, scalability, security, performance, inter-operability, modifiability, maintainability, usability and manageability

Using a different AF to design a system would result in similar but different outcomes because each framework has different *viewpoints* to architecture. On the other hand, using the same AF to design systems with varying complexities would require different types of inputs and would produce different outcomes. Outcomes of an AF reflect the goals a framework sets to achieve. In order to be framework neutral, the use of framework specific terminology to represent *outcomes* is avoided.

- Business Model – describes business models, business requirements, business process, system roles, policy statements
- System Model – models major components of the system. To arrive at a system architecture model, major tradeoffs and design decisions are made. Future system enhancements are also taken into consideration
- Information Model – contains data model, data transformation and data interface

- Computation Model – contains system functional description, system process flow, system operations, software components and interactions
- Software Configuration Model – describes how software is packaged, stored, configured, managed and shared
- Software Processing Model – describes how software processes, software threads and run-time environment are structured
- Implementation Model – describes physical system structure such as operating environment, hardware components and networking components of the system. Models implementation processes such as installation, deployment, configuration and management
- Platforms – describe platform software such as operating systems, hardware and networking components, protocols and standards
- Non-functional Requirements Design – models the structure of the system to reflect design of non-functional requirements
- Transitional Design – provides designs and plans to support system transition and evolution
- Design Rationale – documents reasons of design based on analysis and tradeoffs that involve multiple dimensions of inputs

This paper is not concerned with the format or notation used by AF. Some AF is non-specific on representations of its views, but other frameworks prescribe use of formal description languages.

4. Architecture Frameworks Analysis

This section provides a high level comparison and analysis of six architecture frameworks. Since AFs have different viewpoints or perspectives on how architecture model should be represented, they can be compared only when frameworks are characterized by fundamental elements such as their *goals*, *inputs* and *outcomes*.

Table 1 provides an overview and comparisons of frameworks. If a framework explicitly supports an element in the table, it is reported as "Y". If a framework does not support an element or there is no mention of that element in the documentation, then it is reported as "N". Where a framework partially supports or eludes to support an element, it is reported as partial, "P". The extent to which each framework supports and interpret an element may differ even when they have the same values in the same row. Discussions of individual frameworks with reference to Table 1 are made in each of the following sub-sections.

	ZF	4+1 View	FEAF	RM-ODP	TOGAF	DoDAF
Goals						
Architecture Definition and Understanding	P	P	Y	Y	Y	Y
Architecture Process	N	N	Y	N	Y	Y
Architecture Evolution Support	N	N	Y	P	Y	Y
Architecture Analysis	Y	Y	Y	Y	Y	Y
Architecture Models	Y	Y	Y	Y	Y	Y
Design Tradeoffs	P	P	P	P	P	Y
Design Rationale	P	P	P	Y	Y	P
Standardization	N	N	P	Y	Y	Y
Architecture Knowledge Base	N	N	Y	Y	Y	Y
Architecture Verifiability	N	P	N	P	Y	N
Inputs						
Business Drivers	P	P	Y	P	Y	Y
Technology Inputs	N	N	Y	P	Y	Y
Business Requirements	Y	Y	Y	Y	Y	Y
Information System Environment	P	P	Y	Y	Y	Y
Current Architecture	P	Y	Y	Y	Y	Y
Non Functional Requirements	P	Y	P	Y	Y	P
Outcomes						
Business Model	Y	P	Y	Y	Y	Y
System Model	Y	Y	Y	Y	Y	Y
Information Model	Y	Y	Y	Y	Y	Y
Computation Model	Y	Y	Y	Y	Y	Y
Software Configuration Model	N	Y	N	P	Y	N
Software Processing Model	Y	Y	Y	Y	Y	Y
Implementation Model	P	P	P	Y	Y	Y
Platforms	Y	P	Y	Y	Y	Y
Non-functional Requirements Design	P	Y	P	Y	Y	P
Transitional Design	N	N	Y	N	Y	Y
Design Rationale	N	P	N	P	P	P

Table 1: Comparisons of Architecture Frameworks

4.1. Zachman Framework for Enterprise Architecture (ZF)

The Zachman Framework for Enterprise Architecture is based on Information System Architecture (ISA) and Extended Information System Architecture (EISA) proposed by Zachman. ZF has been widely adopted by the architecture community and it is incorporated into other AFs. ZF's key goals are for enterprise architecture analysis and modeling and it is also concerned with *perspectives* of constructing an information system. A *perspective* is a row in a table representing how a stakeholder in a project team would view the system. The various stakeholders are *Planner, Owner, Designer, Builder and Subcontractor*. Each *perspective* would produce their respective outcomes such as Scope Document, Enterprise or Business Model, System Model, Technology Model and Components. The framework specifies, for each perspective, different types of information that are characterized by (a) what – information and data; (b) how – function and process; (c) where – location of hardware / software; (d) who – people in terms of allocation of work and authority; (e) when – timing requirements of business process; (f) why – motivation.

ZF identifies different stakeholders of a project by perspectives and six aspects, or columns, to characterize information required. The rows and columns intersect to define an architecture design element in a cell. A cell is an *outcome* of an architecture activity based on an aspect of a system for a particular group of people.

ZF provides a concise way to structure and model enterprise and system architecture. Each cell has a singular focus on one aspect of the architecture such as data, process or location. ZF has been referred to and used in frameworks such as FEAF, DoDAF and TOGAF. However, ZF does not prescribe design tradeoffs or design rationale documentation. The framework does not explicitly prescribe support for non-functional requirements or architecture evolution. There is no distinction between architecture modeling activities and detailed design activities in this framework. Unlike TOGAF or DoDAF, ZF only

provides brief descriptions of architectural outcomes and no description on architectural process.

4.2. 4+1 View Model of Architecture

The goals of *4+1 View Model* is for architecture analysis and modeling of software systems. The framework uses four viewpoints to represent architecture models and a scenario view for discovery and verification.

- Logical View – represents the functional requirements of the system
- Process View – this view facilitates partitioning of software into independent software tasks that represent running processes and their inter-process communication in a distributed environment, taking into account non-functional requirements
- Development View – this view focuses on the organization of software modules
- Physical View – this view denotes mapping of software to hardware nodes
- Scenarios – scenarios or instances of use cases are used to discover and test the architecture design

The *4+1 View Model* proposes an iterative approach of architecture design through analysis and decomposition of design issues. The power of this model is to focus on key development issues. Outcomes of the model are represented by UML notation. Similar to RM-ODP, this is a development framework with a primary goal to support development of distributed systems. *4+1 View Model* does not address issues surrounding enterprise architecture. Although design rationale and risk assessment issues are mentioned, how they could be documented in the model is unclear. The framework does not distinguish architecture design from detailed designs.

4.3. Federal Enterprise Architecture Framework (FEAF)

FEAF is a framework issued by the US CIO Council to promote shared development for common US Federal processes, interoperability, and sharing of information among Federal Agencies and other Government entities. The framework is organized in 4 levels. *Level I* is the highest level view which deals with *architecture drivers* or external stimulus and strategic direction of architecture. It facilitates the transformation of current architecture to target architecture through applying architecture standards and managing the architecture process. *Level II* provides more details by analyzing the *business drivers* and *design drivers* of an architecture. The outcome of

this process is target business architecture and target design architecture. *Level III* expresses the architecture in more details by using business, data, applications and technology views to model the target architecture. *Level IV* uses a combination of ZF and Spewak's Enterprise Architecture Planning (EAP) methods. ZF columns of data, functions and network are used to represent *Data Architecture*, *Application Architecture* and *Technology Architecture*.

FEAF supports most of the goals and outcomes described in this paper but it is primarily a framework for architecture planning. At the higher levels, target architecture is expressed in terms of meeting strategic directions, supporting business model and arriving at data architecture, applications architecture and technology architecture. Similar to TOGAF, FEAF uses architecture drivers, business drivers and design drivers as inputs for high level architectural planning. FEAF supports architecture transitions or evolution. Apart from system security, FEAF does not explicitly support other non-functional requirements. Design rationale is not fully considered in this framework.

4.4. Open Distributed Processing – Reference Model (RM-ODP)

The ISO RM-ODP Standards is a set of international standards with four parts. Part 1 (ISO 10746-1/ITU-T X.901) provides an overview and a guide to the use of the reference model. Part 2 and Part 3 (ISO 10746-2/ITU-T X.902 and ISO 10746-3/ITU-T X.903) provide a foundation of concepts and prescribe concepts, rules and functions for the modeling of ODP systems. Part 4 (ISO 10746-4/ITU-T X.904) is the architectural semantics which provides a formal description technique for Part 2 and Part 3. The primary objective is to allow the benefits of distribution of information processing services to be realized in an environment of heterogeneous IT resources and multiple organization domains.

RM-ODP uses five viewpoints to represent different aspects of a system. The *Enterprise Viewpoint* states high level enterprise requirements such as (a) Purpose and objectives of systems, (b) Community or users of system and (c) Business policies, guidelines, flows and constraints and (d) Actions performed. The *Information Viewpoint* focuses on information semantics and information structures. The *Computational Viewpoint* focuses on decomposition of the system and on the constraints of the objects and their interactions. The objects specified and modeled can be computational, service support or infrastructure objects. Interactions between

objects are connected through interfaces. The *Engineering Viewpoint* focuses on mechanisms and functions that support interactions between distributed objects. The *Technology Viewpoint* specifies the choice of technology, including products, standards and technology objects, selected to support the implementation. RM-ODP provides standards to define *transparencies* for the support of distributed processing. *Transparencies* are architecture patterns that are defined in the *Engineering Viewpoint* for hiding transparent functions.

RM-ODP primarily focuses on ODP architecture development. Architecture rationale and tradeoffs are not documented as part of the model. RM-ODP does not provide software configuration model to represent software packaging although *Engineering Viewpoint* may be used to depict it. It does not concern with business strategies or the evolution of the architecture to meet future needs. RM-ODP is formal and it provides a complete and consistent model for the specification of system architecture design. RM-ODP does not prescribe an architecture process and it is non-specific on what level of details architecture modeling require.

4.5. The Open Group Architecture Framework (TOGAF)

TOGAF's goals are to provide a framework for the design, evaluation and building of architectures for enterprises. A key element of TOGAF is TOGAF Architecture Development Method (ADM) that specifies a process for developing enterprise architecture. The *Enterprise Continuum* is a virtual repository of all architecture assets that include models, patterns and architecture descriptions. The *TOGAF Resource Base* is a set of resources, guidelines, templates and background information to assist in the use of TOGAF.

TOGAF ADM is a generic method which specifies an iterative approach for architecture development. ADM is not prescriptive on breadth of coverage, level of details, extent of time horizon or architectural assets to be leveraged. These can be determined by architects to suit a particular project. The phases defined by ADM are the following.

- Preliminary Framework and Principles to define the baseline of architecture within an enterprise
- ADM Cycle defines the architecture development cycle
- Requirements Management process is central to the ADM Cycle where it identifies, stores and

interfaces requirements with all phases of the ADM Cycle.

The TOGAF Enterprise Continuum specifies a Technical Reference Model (TRM). TRM is a model that represents a system in terms of Application, Application Platform and Communication Infrastructure and their inter-connectivity. TRM also describes Service Qualities provided by the system.

TOGAF ADM is a comprehensive methodology that addresses architecture at the enterprise level as well as the individual system level. Its methodology supports architecture evolution through using *Enterprise Continuum* as its knowledge base. Activities in each phase of the ADM framework are well defined but it leaves implementation flexibility to practicing architects to determine what is required for the system from a defined set of possible outcomes. TOGAF recommends documentation of design rationale to trace design and architecture decisions.

4.6. Department of Defense Architecture Framework (DoDAF)

The Department of Defense (DoD) Architecture Framework Version 1.0 is developed specifically for the US DoD to support its war-fighting operations, business operations and processes. It grew from and superseded the previous AF, C4ISR Architecture Framework Version 2.0. Architecture development techniques have been provided in DoDAF to specify processes for scope definition, data requirements definition, data collection, architecture objectives analysis and documentation.

DoDAF uses Core Architecture Data Model (CADM) for architecture documentation. CADM is a standardized taxonomy to define views and their elements in a database. *All Views* provide an overview, summary and integrated dictionary of the architecture; *Operational Views* describe the business and operation of the architecture, they describe operation nodes, nodes connectivity, information exchange, organization relationship, operation rules, event-trace and logical data model; *System Views* describe the system and its components; *Technical Views* describes the current standard profile and future technical standards forecast.

The DoDAF framework is specifically designed to support defense operations and therefore some of its processes and taxonomies are domain dependent. CADM is a well defined schema to support the documentation of architecture models in this domain. Using CADM and traceability matrix, operational requirements and design decisions can be traced in the

architecture. Although DoDAF provides traceability, it does not have provision to record architecture rationale. DoDAF does not provide modeling capability for software configuration and it offers limited support for modeling of non-functional requirements.

5. Discussion

This study uses *goals*, *inputs* and *outcomes* as fundamental elements to analyze AF. It shows that all architecture frameworks support the purpose of software architecture development. In particular, RM-ODP and *4+1 View Model* both have a singular focus on software architecture development. The common framework characteristics for software architecture development can be typified by the elements they support, as indicated by a 'Y', in Table 1: (a) *Goals* - Architecture Analysis and Architecture Models; (b) *Inputs* – Business Requirements, Information System Environment and Current Architecture; (c) *Outcomes* – Business Model, System Model, Information Model, Computation Model, Software Processing Model and Platforms. The use of common viewpoints provides a good representation of software development frameworks: business requirements (though not business strategies) in the enterprise/business viewpoint; data modeling in the information viewpoint; software architecture in the computation viewpoint; processing architecture in the engineering viewpoint and platforms in the physical/technology Viewpoint.

TOGAF, DoDAF and FEAF address enterprise architecture issues such as architecture planning, evolution and system interoperability. They use different views for enterprise architecture modeling and have different degrees of specificity in their views. It is clear from Table 1 that enterprise architecture is characterized by their support of the following common elements: (a) *Goals* - Architecture Definition and Understanding, Architecture Process, Architecture Evolution Support, Standardization and Architecture Knowledge Base; (b) *Inputs* – Business Drivers and Technology Inputs; (c) *Outcomes* – Business Model and Transitional Design. ZF is an enterprise framework but the lack of detailed description of the framework makes it difficult to further analyze its capabilities in this respect. The focus of enterprise frameworks is to facilitate the definition, common understanding and standardization of architecture practice in an enterprise. Their long term goals are to support strategic architecture planning, use an architecture knowledge base to support architecture

evolution. The business and architecture models produced from these frameworks describe architecture directions, the "to-be" architectures, and the enterprise strategies to transition from the current architecture to the future architecture.

It is found that there are two major deficiencies of AFs. (a) The level of details required in an architecture model is generally not specified. (b) Architecture rationales are not a mandatory part of the model. The implication is that architecture models cannot be verified or traced. This is evident by the lack of support in Design Tradeoffs, Design Rationale and Architecture Verifiability in Table 1.

Although architecture involves design, the objective of architecture differs from detailed design. Design activities are concerned with conceiving and designing in a focused area where architecture is concerned with structure, modeling and planning of the system at a higher level. There are no guidelines in any frameworks to distinguish between architecture activities and the extent to which they become detailed design activities. We propose that the guiding principle of the level of details of design in architecture is based on the level of risk. If the risk, or uncertainty, of the architecture to accurately model the system is relatively small, then design could be carried out in the detailed design phase. On the other hand, if the uncertainty or the risk is high, then more architecture activity is required to develop the design to reduce its risk.

All architecture frameworks surveyed either omit or have very little description of architecture design rationale even though they are crucial to a system. Architectures have to be correctly designed and verified at the early stages of the system. We propose to incorporate design rationale in architecture frameworks with the following features: (a) cross-reference *requirements* to *architecture design* for consistency checking and traceability [14]; (b) document tradeoffs rationale based on quantification of costs, benefits and risks; (c) use scenarios to depict design analysis [8]; (d) describe compromises and enhancements made to requirements; (e) describe feasibility and infeasibility of the proposed design. For a given set of inputs, there would be more than one possible architecture design choices. Each design choice at a point of decision is associated with a set of costs, benefits and risks which are measured both quantitatively and qualitatively. Costs do not only represent the resources required to realize a design but also represent the compromise, or opportunity costs, in terms of functionality and other factors. Similarly, benefits represent the relative benefits of a design. Risks represent the level of uncertainty to realize

desired business objectives due to technical, business or other resource reasons.

Architecture designs require a set of multi-dimensional inputs 1 to j as design considerations. Each architecture design option, from 1 to i, is associated with a set of costs, benefits and risks. *Architecture$_i$* is the result of the function *ArchitectureDesign$_i$* as shown below.

$$ArchitectureDesign_{[1..i]} (Input_1 \dots Input_j) \rightarrow$$
$$Architecture_{[1..i]} [Costs, Benefits, Risks]$$

The *art* and *science* of architecture is to choose an architecture design from all possible design choices that best satisfies the goals of a system given a balanced view of minimizing risks and costs whilst maximizing benefits. This is the architecture tradeoffs process. Currently some architecture frameworks expressed tradeoffs rationale by documenting system limitations and assumptions. We believe architecture frameworks should be augmented to document rationales which include costs, benefits and risks.

Further details on architecture rationale and the use of risk analysis in architecture design will be reported in a separate paper.

6. Conclusions

This paper has presented a comparative analysis of architecture frameworks. In this regards, we have introduced a model of understanding for AF based on fundamental elements of architecture. With this model of understanding, AF could be selected or tailored for system or enterprise architecture development in specific environments. Selected elements from multiple frameworks could be used in conjunction to meet particular development needs.

To analyze frameworks that have varied viewpoints, we group fundamental elements into *goals*, *inputs* and *outcomes* to enable analysis. Based on architecture frameworks' support of these elements, we found two classes of frameworks with distinct characteristics: Software Architecture Framework and Enterprise Architecture Framework. All frameworks surveyed here support software architecture development but only three frameworks support enterprise architecture modeling.

We have identified several common deficiencies of architecture frameworks especially in the area of architecture rationalization. As such, we put forward a notion of using costs, benefits and risks as a foundation for tradeoffs. Design rationale can be used for associating architecture designs to provide

traceability and verifiability. No surveyed frameworks specify their requirements on the level of details of architecture design. We believe that this distinction is important from the development life cycle viewpoint. As such, we suggest using architecture risk analysis to determine how much architecture design is required.

7. References

[1] D. Garlan and M. Shaw, "An Introduction to Software Architecture," *Advances in Software Engineering and Knowledge Engineering*, vol. 2, pp. 1- 39, 1993.

[2] D. E. Perry and A. L. Wolf, "Foundation for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, pp. 40- 52, 1992.

[3] R. T. Monroe, A. Kompanek, R. Melton, and D. Garlan, "Architectural Styles, Design Patterns, and Objects," *IEEE Software*, vol. 14, pp. 43- 52, 1997.

[4] IEEE, "IEEE Recommended Practice for Architecture Description of Software-Intensive System (IEEE Std 1471-2000)," IEEE Computer Society 2000.

[5] A. Eden and R. Kazman, "Architecture, Design, Implementation," presented at International Conference for Software Engineering, 2003.

[6] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Boston: Addison Wesley, 2003.

[7] J. Zachman, "A framework for Information Architecture," *IBM Systems Journal*, vol. 38, 1987.

[8] P. Kruchten, "The 4+1 View Model of Architecture," *IEEE Software*, vol. 12, pp. pp 42-50, 1995.

[9] CIO-Council, "Federal Enterprise Architecture Framework version 1.1," 1999, http://www.cio.gov/archive/fedarch1.pdf.

[10] ISO/ITU-T, "Reference Model for Open Distributed Processing (ISO/ITU-T 10746 Part 1 - 4)," Information Standards Organisation 1997.

[11] The Open Group, "The Open Group Architecture Framework (ver 8.1 Enterprise Edition)," 2003, http://www.opengroup.org/architecture/togaf/#download.

[12] Department of Defense, "Department of Defense Architecture Framework Version 1.0 - Vol 1 Definition & Guideline and Vol 2 Product Descriptions," 2003, http://www.aitcnet.org/dodfw.

[13] J. Han and P. Chen, "Architecture Support for System-of-Systems Evolution," presented at First International Conference, EDCIS 2002 Proceedings, 2002.

[14] J. Han, "TRAM: A Tool for Requirements and Architecture Management," presented at Proceedings of the 24th Australasian Computer Science Conference, Gold Coast, Australia, 2001.