School of Information Technology
Centre for Component Software and Enterprise Systems

A Comparative Analysis of Architecture Frameworks

Technical Report: SUTIT-TR2004.01
CeCSES Centre Report: SUT.CeCSES-TR001

Antony Tang, Swinburne University of Technology
Jun Han, Swinburne University of Technology
Pin Chen, DSTO, Department of Defence
25 August 2004

.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Table of Contents

School of Information Technology
Centre for Component Software and Enterprise Systems

A Comparative Analysis of Architecture Frameworks

Abstract

Architecture frameworks are methods used in architecture modelling. 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 analysing 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.

Keywords
Architecture Frameworks, Information Systems, Viewpoints, Design, Tradeoffs, Rationale, Risks

Introduction

This paper investigates the concept of architecture by examining six architecture frameworks. Architecture plays a major role in the development of information systems. Typically, the development of an information system would entail requirements gathering, analysis, software design, development and hardware configuration. The act of architecture in this 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 (Garlan and Shaw 1993) suggested that software architecture is concerned with issues beyond algorithms and data structures of computation. Perry and Wolf (Perry and Wolf 1992) 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. (Monroe et al. 1997) suggested that architecture is not about details of implementation. Architecture Practice was proposed to bring separate or isolated architectures and activities into an engineering context (Chen and Pozgay 2002). 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" (IEEE 2000). There was also an attempt to formally distinguish architecture activities from design activities (Eden and Kazman 2003).

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 modelling, 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?

With new developments in distributed applications and technologies, software is only one of the components in an information system. Further design is required in diverse areas of requirements modelling, network infrastructure, servers configuration and middleware technologies. For instance, the security design of a system may require security process engineering, design of multiple layers of firewalls and network configuration to complement software security. An architecture framework would provide a consistent approach for standardising, planning, analysing and modelling of these components.

Several Architecture Frameworks have been published for this purpose. Some were published by organisations to guide their internal system development. Others were published by standards body for establishing architecture standards. Activities defined in these architecture frameworks vary and so do their outcomes. By analysing and comparing architecture frameworks, this paper provides a model of understanding for architecture. We attempt to identify any deficiencies in these architecture frameworks and propose improvements to overcome them. The section on Architecture Frameworks introduces what they intend to achieve. The Basis for Analysis defines a neutral way to study the frameworks. The section on Architecture Frameworks Analysis presents the study of six architecture frameworks. The Discussion section analyses and compares frameworks, their deficiencies and proposes solutions. We conclude our findings in the last section.

Architecture Frameworks

There is no universally agreed process and representation of architecture amongst researchers and practitioners. Architecture modelling commonly uses high level abstraction called views. Architecture frameworks 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, the IEEE Recommended Practice for Architectural Description of Software-Intensive System (IEEE 2000) and methods for architecture evaluation (Barbar et al. 2004), this paper puts forward a set of common goals for architecture frameworks. 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 a framework that supports 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” (Bass et al. 2003)
- Standardisation – 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 architecture frameworks. As such, resulting architecture models produced by architecture frameworks would provide consistent representation of architecture models. A well defined architecture process would provide a structured approach to architecture analysis and modelling for making design decisions based on tradeoffs. Architecture rationale cross-reference design decisions to facilitate verification of architecture models. Architecture models are documented and stored in a standard architecture knowledge base to support change management and system evolution.

There are a number of established architecture frameworks. The Zachman Framework for Enterprise Architecture (Zachman 1987, Sowa and Zachman 1992, Zachman 1996) is one of the earliest works in this area. *4+1 View Model of Architecture* is a framework for modelling software architecture (Kruchten 1995). The US Federal Government issued a standard Federal Enterprise Architecture Framework version 1.1 (CIO-Council 1999) which aims at maximising the benefits of information technology within the US Government. The International Standards Organisation (ISO) in conjunction with the International Telecommunication Union (ITU-T) issued a Reference Model for Open Distributed Computing (RM-ODP). The intention of this model is to define and integrate a wide range of ODP standards for distributed systems (ISO/ITU-T 1997). The Open Group, an industry standard organisation, issued The Open Group Architectural Framework (TOGAF) version 8.1 in 2003. TOGAF is a method and a set of supporting tools for developing enterprise architecture (The Open Group 2003). The US Department of Defense released the DoD Architecture Framework version 1.0 (Department of Defense 2003) for all architectures developed within the DoD to comply with. The selection of architecture frameworks 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 architecture framework but to analyse and compare their similarities and differences.

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. (Han and Chen 2002). The dimensions, or inputs, to the architecture process are themselves interrelated and therefore they cannot be considered in isolation in the architecture 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 minimised. Risks that arise from construction or evolution of an information system may lie in many areas. They represent the uncertainty of achieving the system objectives.

Architecture activities remove major uncertainties through modelling and specification of the system to the point where the problem to be solved become well understood and the modelled solution has a high certainty of achieving its objectives.

In this paper, we propose to analyse architecture frameworks in terms of their goals, inputs and outcomes. The goals are described in the previous section. The inputs represent inputs that architecture modelling activities could use. The outcomes represent results and deliverables from using architecture frameworks. The 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, organisation 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 architecture framework 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 architecture framework to design systems with varying complexities would require different types of inputs and would produce different outcomes. Outcomes of an architecture framework reflect the goals a framework sets to achieve. This paper uses a list of common outcomes of architecture frameworks to demonstrate their characteristics. In order to be framework neutral, the use of framework specific terminology 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. For instance, to design for reliability requirements require modelling and design in all of the following areas in a collaborative manner: (a) software replication; (b) software switch over; (c) transaction consistency; (d) database recovery and (e) hardware data protection.
- Transitional Design – provides designs and plans to support system transition and evolution
- Design Rationale – documents reasons of design based on analysis and tradeoffs of multiple dimensions of inputs

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

Architecture Frameworks Analysis

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

	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

Table 1 provides an overview and comparisons of frameworks. If an architecture 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 an architecture 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 each framework are made in the following sections where their general goals, inputs and outcomes are analysed and compared.

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 architecture frameworks. ZF's key goals are for enterprise architecture analysis and modelling and it is concerned with *perspective*s 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 characterised 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. A summary of this framework is listed in table 2.

	Data (what)	**Function (how)**	**Network (where)**	**People (who)**	**Time (when)**	**Motivation (why)**
Scope (Planner)	List of things important to business	List of processes the business performs	List of locations which the business operates	List of organisations / agents that are important	List of significant events	List of business goals / strategies
Enterprise Model (Owner)	Semantic Model	Business Process Model	Business Logistic System	Work Flow Model	Master Schedule	Business Plan
System Model (Designer)	Logical Data Model	Application Architecture	Distributed System Architecture	Human Interface Architecture	Processing Structure	Business Rules
Technology Model (Builder)	Physical Data Model	Systems Design	Technology Architecture	Presentation Architecture	Control Structure	Rule Design
Components (Subcontractor)	Data Definition	Program	Network Architecture	Security Architecture	Timing Definition	Rule Specification

Table 2: Zachman Framework for Enterprise Architecture

The framework identifies different stakeholders of a project, by *perspectives*, and for each of the six aspects or columns, defines 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, design rationale or documentation of architecture decisions. The framework does not explicitly prescribe support for non-functional requirements or architecture evolution. There is no distinction between architecture modelling 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+1 View Model of Architecture

The goals of *4+1 View Model* is for architecture analysis and modelling 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. Functions are decomposed and abstracted using object classes and objects. The notation recommended is UML
- 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. It also represents processing requirements such as concurrency and distribution, integrity and fault-tolerance
- Development View – this view focuses on the organisation of software modules. The main concerns are ease of development, software management, software reuse and software environmental constraints
- Physical View – this view denotes mapping of software to hardware nodes. It takes into consideration system availability, reliability, performance and scalability
- 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 into smaller design issues. The power of this model lies in its elegance to focus on key development issues. Outcomes of the model are represented using UML notation which can easily be decomposed into detailed design. Similar to RM-ODP, this is a development architecture framework with a primary goal to support software architecture development of distributed systems. 4+1 View does not address issues such as system evolution and issues surrounding enterprise architecture. Although design rationale and risk assessment issues are mentioned, how they could be documented in the model is unclear. This framework has no discussion on the level of details architecture modelling should go into to sufficiently represent the architecture design.

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 objective of CIO Council is to have a standard framework for improving practices in the design, modernisation, use, sharing and performance of Federal information resources. The framework is organised 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 analysing 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*. All five perspectives of the ZF framework are employed. EAP is used to define the process of architecture planning.

FEAF supports most of the goals and outcomes described in this paper but it is primarily a framework for architecture planning. . A complementary Practical Guide to FEAF defines the process for applying FEAF (CIO-Council 2001). 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 through transitional processes. Apart from system security, FEAF does not explicitly support other non-functional requirements. Design rationale is not fully considered in this framework. It does not support software configuration modelling.

Open Distributed Processing – Reference Model (RM-ODP)

The ISO RM-ODP Standards is a set of international standards. RM-ODP standards have 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 modelling 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 of the standard is to allow the benefits of distribution of information processing services to be realised in an environment of heterogeneous IT resources and multiple organisation domains. ODP standardisation considers distributed systems spanning many organisations and technological boundaries. These systems can be heterogeneous, autonomous, evolving and mobile in nature.

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 outcome of the enterprise viewpoint can be represented by semi-formal documentation using text and UML diagrams. The *Information Viewpoint* focuses on information semantics and information structures. It does not capture information flow (this is done via *Computational Viewpoint*) but instead specifies the schema of the information in its specification. The *Computational Viewpoint* focuses on decomposition of the system and on the constraints of the objects and their interactions. The objects specified and modelled can be computational, service support or infrastructure objects. Interactions between objects are connected through *interfaces*. Interfaces can be classified into three types: signal, operation and flow. Each interface type specifies how two objects interact with each other and what interface signature is used in that interaction. The *Engineering Viewpoint* focuses on mechanisms and functions that support interactions between distributed objects. *Basic engineering objects* (BEO) that correspond to objects from computational specification are grouped together by cluster, capsule and node to reflect their relationship with the physical environment. An *interconnection model* is also provided to define the connectivity between

BEOs in various nodes. 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 abstract designs or architecture patterns that are defined in the *Engineering Viewpoint* for hiding transparent functions. The eight transparencies are Access, Failure, Location, Migration, Persistence, Relocation, Replication and Transaction. RM-ODP uses consistency checks to verify the consistency of the model across multiple views. The consistency check examines the correspondence of objects across different viewpoints. Architecture rationale and tradeoffs are not documented as part of the model. Scenarios based architecture design process is not used in RM-ODP. RM-ODP does not provide software configuration model to represent software packaging although *Engineering Viewpoint* may be used to depict it.

Unlike FEAF or TOGAF, which prescribe architecture planning and review, RM-ODP is a framework for the modelling and development of ODP. It does not concern with future 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. This is an advantage over some of the other approaches because it provides consistency across views (Putman 2001). RM-ODP does not prescribe an architecture process and it is non-specific on what level of details architecture modelling require.

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 – this phase should define the baseline of architecture within an enterprise through defining the framework and defining the architecture principles
- ADM Cycle : (A) *Architecture Vision* includes high-level description of the baseline of current ("as-is") and desired ("to-be") architectures from both business and technical perspectives; (B) *Business Architecture* describes the baseline business architecture and analyse gaps between baseline and target business architecture; (C) *Information System Architecture* describes the target data and application architecture by analysing the data and application requirements; (D) *Technology Architecture* is used to develop the architecture that will form the basis for implementation. There are eight sub-phases in this step including the creation of baseline, considering views, creating architecture model, selecting services, confirming business objectives, determining criteria, defining architecture and conducting gap analysis; (E) *Opportunities and Solutions* phase involves the evaluation and selection of implementation options; (F) Migration

Planning is concerned with prioritising projects implementation based on their dependencies; (G) *Implementation Governance* is concerned with the architecture contract that governs the overall implementation and deployment of a project; (H) *Architecture Change Management* is the continual monitoring of changes in new technology and business environment that could instigate new developments.

- 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. The Standard Information Base in the Enterprise Continuum is a database of facts, guidance and information systems standards. Part of the TOGAF Resource Base provides a guideline to developing architecture views. TOGAF recommends the use of Business Architecture View, Data Architecture View, Application Architecture View, Technology Architecture View, System Engineering View, Enterprise Security View, Enterprise Manageability View, Enterprise Quality of Service View and Enterprise Mobility View. TOGAF supports non-functional requirements through Quality of Service views.

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 which could be used to trace design and architecture decisions.

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 superceded the previous architecture framework, C4ISR Architecture Framework Version 2.0. DoDAF includes guidelines on determining architecture content based on intended use; focus on using architectures in support of DoD's Programming, Budgeting, and Execution process; Joint Capabilities Integration and Development System; and the Defense Acquisition System; and increasing emphasis on the architecture data elements. 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 prescribes the documentation of architecture through the use of Core Architecture Data Model (CADM) and architecture products. CADM is a standardised taxonomy to define views and their elements in a database. A view would represent *operations*, *systems* or *technical standards* of the architecture. Each view has a well-defined set of data elements in CADM that represents the context of the view. Architecture products are documentation that describes the architecture views using text and UML as modelling language. *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, organisation relationship, operation rules, event-trace and logical data model; *System Views* describe the system and its components, it describes system interface, communication, system-system matrix, system functionality, operation to system traceability matrix, system evolution, performance and technology forecast; *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 modelling capability for software configuration and it offers limited support for modelling of non-functional requirements.

Discussion

This study compares and analyses six architecture frameworks. To enable the analysis, framework specific features such as domain and notation are not compared. Rather the goals, inputs and outcomes are used as fundamental elements in the analysis. 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 characteristics of frameworks for software architecture development can be typified by their support of elements, 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) O*utcomes* – System Model, Information Model, Computation Model, Software Processing Model and Platforms.

TOGAF, DoDAF and FEAF address enterprise architecture issues such as architecture planning, evolution and system interoperability. They use different views for enterprise architecture modelling and have different degrees of specificity in their views. They use different views for modelling and have different degrees of specificity in their views. It is clear from Table 1 that enterprise architecture is characterised by their support of the following common elements: (a) *Goals* - Architecture Definition and Understanding, Architecture Process, Architecture Evolution Support, Standardisation 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 analyse its capabilities in this respect.

In the analysis, it is found that there are common deficiencies of architecture frameworks. (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. (c) Specification of non-functional requirements is lacking from some of the architecture frameworks. Modelling of non-functional requirements usually involves architecture design in multiple views. They have significant costs and risks impact in a complex system environment and therefore architecture rationale is of high importance here. (d) Software configuration modelling is generally lacking from some of the software architecture frameworks.

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, modelling 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. It is difficult to distinguish precisely what should be performed in architecture or detailed design and what role an architect or a designer should play. As such, this paper proposes 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. When a well known architecture pattern or design pattern is chosen to solve a well understood business requirement and the risk to the system is judged to be small, then the goal of architecture is fulfilled. 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. For instance, if performance of a system is of importance and it is anticipated that the current technology might not cope with the desired transaction throughput, then more architecture design effort is required in the modelling phase to investigate into software design, system design and other areas to reduce the risk and ascertain that the design would work with respect to performance requirements.

Design rationales need to be documented and associated with architecture models for traceability and verifiability. All architecture frameworks surveyed either omit or have very little description of architecture design rationale. We propose that architecture design rationale should include the following features: (a) cross-reference *requirements* to *architecture design* for consistency checking and traceability (Han 2001); (b) document tradeoffs rationale based on quantification of costs, benefits and risks; (c) use scenarios to depict design analysis (Kruchten 1996); (d) describe compromises and enhancements made to requirements; (e) describe feasibility and infeasibility of the proposed design.

Architecture is a process of considering all relevant inputs, identifying design choices, considering tradeoffs and finally selecting an appropriate design. For a given set of inputs, there would be more than one possible architecture design choices. Each design choice 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 realise a design but also represents the compromise, or opportunity costs, in terms of functionality and other factors that are associated with the design in the model. Similarly, benefits represent the relative benefits of a design. Risks represent the level of uncertainty to realise 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 different set of costs, benefits and risks. $Architecture_i$ is a result of the function $ArchitectureDesign_i$ as shown below.

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

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 that architecture frameworks should be augmented to document tradeoffs rationale to include costs, benefits and risks.

Conclusions

This paper has presented an analytical study of various architecture frameworks. In this regards, we have introduced a model of understanding for architecture frameworks based on fundamental elements of architecture. With this model of understanding, architecture frameworks 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 analyse 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 modelling.

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 based on tradeoffs 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.

References

Barbar, M A, et al. (2004) A Framework for Classifying and Comparing Software Architecture Evaluation Methods, *Proceedings 2004 Australian Software Engineering Conference*, 309-318.

Bass, L, et al. (2003), *Software Architecture in Practice,* Addison Wesley, Boston.

Chen, P & Pozgay, A (2002) Architecture Practice: A Fundamental Discipline for Information Systems, *Australian Conference on Information Systems 2002*, Melbourne, Australia,

CIO-Council (1999) Federal Enterprise Architecture Framework version 1.1, URL http://www.cio.gov/archive/fedarch1.pdf, Accessed 21 May 2004.

CIO-Council (2001) A Practical Guide to Federal Enterprise Architecture version 1.0, URL http://www.cio.gov/archive/bpeaguide.pdf, Accessed 21 May 2004.

Department of Defense (2003) Department of Defense Architecture Framework Version 1.0 - Vol 1 Definition & Guideline and Vol 2 Product Descriptions, URL http://www.aitcnet.org/dodfw, Accessed 1 April 2004.

Eden, A & Kazman, R (2003) Architecture, Design, Implementation, *International Conference for Software Engineering*, pp 149 - 159.

Garlan, D & Shaw, M (1993) An Introduction to Software Architecture, *Advances in Software Engineering and Knowledge Engineering,* 2**,** 1- 39.

Han, J (2001) TRAM: A Tool for Requirements and Architecture Management, *Proceedings of the 24th Australasian Computer Science Conference*, Gold Coast, Australia, Jan 2001, IEEE Computer Society Press 60-68.

Han, J & Chen, P (2002) Architecture Support for System-of-Systems Evolution, *First International Conference, EDCIS 2002 Proceedings*, 332-346.

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

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

Kruchten, P (1995) The 4+1 View Model of Architecture, *IEEE Software,* 12**,** 6 pp 42-50.

Kruchten, P (1996) Software Architecture-A Rational Metamodel, *Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops*,

Monroe, R T, et al. (1997) Architectural Styles, Design Patterns, and Objects, *IEEE Software,* 14**,** 1 43- 52.

Perry, D E & Wolf, A L (1992) Foundation for the Study of Software Architecture, *ACM SIGSOFT Software Engineering Notes,* 17**,** 4 40- 52.

Putman, J (2001), *Architecting with RM-ODP,* Prentice Hall, NJ.

Sowa, J & Zachman, J (1992) Extending and formalising the framework for Information Systems Architecture, *IBM Systems Journal,* 31**,** 3.

The Open Group (2003) The Open Group Architecture Framework (ver 8.1 Enterprise Edition), URL http://www.opengroup.org/architecture/togaf/#download, Accessed 21 May 2004.

Zachman, J (1987) A framework for Information Architecture, *IBM Systems Journal,* 38**,** 2&3.

Zachman, J (1996) Enterprise Architecture : The Issue of the Century, URL http://www.zifa.com/zifajz01.htm, Accessed 1 May 2004.