Getting serious about Enterprise Architecture

[22nd EAPC - London]

Bhavish Kumar
Deputy Practice Leader, ASP Europe
bhavish.kumar@cognizant.com

April 2009
Session - Objectives

- A clear understanding of the meaning of:
  - Enterprise architecture in terms of components and inter-relationships
  - Best practices around modelling methods and practices

- An appreciation of:
  - Formal methods of enterprise modelling
  - Testable architectures as an extension to Enterprise architecture methods to help clearly articulate formal descriptions for component inter-relationships
  - Real life implementation and benefits of testable architectures
Agenda

- Setting the scene

- Enterprise Architecture Definition and Modelling Methods
  - TOGAF ADM
  - Archimate

- Introduction of Testable Architecture Methodology
  - Testable Architecture as an extension to Enterprise Architecture

- Customer Case Study
  - Processes, Methods and Tools used
  - Benefits achieved

- Q & A
IEEE Std 1471-2000:
The “architecture” of a system is the system’s **fundamental organization**, embodied in its **components**, their relationships to each other and to the environment, and the **principles** guiding its design and evolution.

The Open Group Architecture Framework (TOGAF version 9):
- A **formal description** of a system, or a detailed plan of the system at **component** level, to guide its implementation (source: ISO / IEC 42010: 2007)
- The structure of **components**, their **inter-relationships**, and the **principles** and guidelines governing their **design and evolution** over time

Other definitions submitted to The Open Group - EA Definition Project:
- **Enterprise Architecture** is a set of **principles**, **practices** and **processes**, that defines the **structure** as well as **operations** of the enterprise and its systems for effective realization of enterprise goals to **enable** an enterprise performance to be **predictable**, **measurable** and **manageable**

- **Enterprise architecture** is a management **discipline** concerned with describing the **components** of an enterprise and the **inter-relationships** between those components necessary to achieve the enterprise’s purpose

- **Enterprise Architecture** is a practice **discipline** characterised by a complete collection of **tools**, **methods**, and **models** to be employed by any enterprise to optimize the business and information assets

- **The EA discipline** defines and maintains the **architecture models**, governance and **transition initiatives** necessary to co-ordinate an organization towards common business and/or IT goals to ensure the enterprise is **fit for purpose** to achieve it’s mission

- **By being inclusive with all other management frameworks, EA is the discipline** that helps the Enterprise define, develop and exploit **boundary-less information flow** capabilities in order to achieve the Enterprise’s Strategic intent
Setting The Scene
- What are we going to focus on?

System's fundamental organization

Effective Integration with Business Partners

Model Driven

Practical Measurable Flexible

Component inter-relationships

Formal grounding

Incremental Development

Standardization

Operational Improvement

Relationship to internal & external environment

Tools & methods

Enterprise Architecture

Testable Architecture

Case Study

©2009 Cognizant Technology Solutions
• **Ambiguity**
  - in requirements (capture, analysis or engineering)
  - between architecture and requirements
  - between implementation and architecture

• Ambiguity exists because requirements are divorced from architecture and architecture from implementation, as a result we end up with:
  - *Poor alignment of IT to business*
  - *High cost in managing complexity*
  - *High cost of testing*
  - *Lack of transparency and control in delivery and change management*
  - *Poor reuse of IT assets*
  - *Lack of business agility hindered by IT*

**Removing ambiguity, joining things up, moves us from “art” to engineering**

**Leading to industrialisation of IT**
Benefits using ADM we have seen:

- **Integration**
  - Integrates with other enterprise architecture processes/frameworks (i.e. Zachmann, Gartner etc)
  - Facilitates integration of enterprise wide processes (i.e. by collecting artefact etc.)

- **Efficiency**
  - Creates a repeatable and predictable process of developing enterprise architecture content
  - Can be extended and customised as per the specific needs of the enterprise for e.g. scaling

- **Simplicity**
  - Process Driven: Inputs, Outputs and Steps are specified for each phase

- **Predictability** of the Outcome
  - The Outputs from one phase could be traced back to the inputs of another phase – i.e. it links inputs to the outcomes

- **Complexity**
  - Not really a bad thing if you learn how to manage the complexity of ADM 😊
Benefits of usage:

- Precise language to document at the enterprise architecture level focusing on structure and semantics
- Enabling Consistent Architecture Communication
- Integrated and Coherent modelling
- Driving Architecture Analysis before actual implementation
- Excellent *High-Level* modelling *within* a domain through visualisation techniques

Source: Telematica Instituut
Models are for humans.

Models are used to create some representation of one or more domain/s.

The level of a model and the semantics of that level are entirely to do with the level of abstraction that we wish to use in order to make the points that need to be made.

Abstraction can be seen as a scoping operator over a domain in which some things are hidden that do nothing to make the points that need to be made.

Models and their levels should be complete and unambiguous with respect to their level.

A model at any level should be able to be type checked and checked for consistency so that it may be said to be correct against that level.

Levels should support operators that enable a full or partial mapping from one level to another.
Say a set of requirements at R0 is said to be met by a model L0 and that model L0 is comprised of several parts (usually aligned to lines of business) then a phased approach could be adopted such as:

ADM drives AS-IS and TO-BE Definitions in:
- Business
- Information
- Technology
Transformation – AS-IS to TO-BE

LOB1
LOB2
LOB3
LOB4
LOB5
LOB6
LOB7

Testing at design time reduces risk of mis-delivery

Generating R4 requirements ensures Alignment of delivery

Testing against R4 ensures Alignment of delivery

generate technical contracts

Review

Background
EA Definition
EA Modelling
Testable Architecture
Case Study
Testable architecture enables the architecture of a system to be described unambiguously such that it may be tested against requirements and used to generate implementation artefacts for delivery thereby improving governance and control.

If we can deliver a solution that connects requirements to architecture and architecture to implementation we shall change the nature of complex solution delivery, reducing costs, risks and time to market in the process.

A lot of disjoint artefacts which breeds ambiguity with no real hope of measuring implementation against specification in any automated computable manner.

Background
EA Definition
EA Modelling
Testable Architecture
Case Study
Introduction of Testable Architecture

**Definition**
An unambiguous formal description of a set of components (CDL) and their ordered interactions coupled with any constraints on their implementation and behaviour (RuleML).

**Formal Grounding**
- Testable architecture has originated from & has strong foundations in “pi-calculus”
  - a formal communication framework developed by Prof. Robin Milner – Professor Emeritus of Computer Science at the University of Cambridge and Turing Award Recipient
- Enables reasoning of descriptions to ensure consistency and correctness against requirements

**Benefits**
- **RISK** → reduced risk of mis-delivery
- **COST** → reduced cost of implementation and testing
- **QUALITY** → increased quality of overall solution
- **AGILITY** → increased agility of overall solution
Repositioning the cost of errors

Testable Architecture is FORMAL hence it reduces defects injection

Fact 1#: Real Implementations using Testable Architecture (CDL)

HL7
• Life sciences principle message interchange standard
• CDL provides the dynamic model for message order enabling rapid deployment of HL7 compliant services (aka SOA)

ISDA
• Derivatives principle message interchange standard
• CDL provides the dynamic model for confirmations, affirmation, etc.
• Enabled rapid compliance to business protocols reducing lifecycle costs

Redhat
• Principle system description providing unique differentiator for Redhat’s SOA platform
• Part of the community edition of Overlord

Fact 2: Redhat

Fact 3: Testable Architecture will reduce cost of a defect, increase profit margin whilst increasing the ROI.

This is a typical Defect Density Graph, that shows the amount of defects found across the phases of a software development life cycle.
Testable Architecture Methodology

- **Gather** requirements
  - UML 2.0 Sequence diagrams & messages
- **Model** (CDM)
  - Systems architecture of interactions
- **Test**
  - J2EE, .NET against model
  - - the model against requirements
- **Implement**
  - J2EE, .NET
- **Guide** Implementation
  - UML, WSDL, BPEL, HTML
  - (AUTO GENERATION)
- **Monitor**
  - Runtime enforcement
- **Verify** model (EXPORT)
  - Sign off on description - BPMN, HTML

Removing Ambiguity means:

- **Driving up quality**
- **Driving down costs**
- **Increasing agility in a controlled manner**
Alignment of Modelling Methods

TOGAF ADM

Enterprise Communications

Business Architecture

Information Systems Architecture

Technology Architecture

Opportunities and Solutions

Implementation Governance

Architecture Change Management

Preliminary

Requirements Management

Archimate

Testable Architecture

Enterprise Structure

Common Vocabulary

AS-IS & TO-BE Modelling

Analysis

Enterprise Communications

Behaviour & Touch points

Interaction Modelling

Testing

Ordered interactions

Formal semantics

Auto-Generate

Detailed Scenario Level Modelling

Use case articulation

BPMN

HTML

STATE MACHINES

WSDL / BPEL

Scale & Formalisms

Testability

Feedback

Influences & Aids

Phases E, G and H

TOGAF ADM

Defines

Touch points & views

Background

EA Definition

EA Modelling

Testable Architecture

Case Study

©2009 Cognizant Technology Solutions Commercial in Confidence
Case Study - Background

Current
- A Major Global Underwriting Business
- Recent M&A issues
- Federated Policy Admin Systems
- Inadequate Process Automation

Business Strategy
- Be Customer Centric
- Diversification to high margin products
- Operational Inefficiencies
- Global Expansion through M&A
- Better integration with partners and across channels

Technology Drivers
- Componentised Core Architecture
- Service Oriented Architecture & BPM
- Digital Asset Management
- Portal Based Solutions – Underwriting and Causality
- Enterprise Data warehouse & BI

IT Strategy
- Speed-to-Market
- Global Platform for Causality – Model office
- E-claims and PAS Consolidation
- Automation of Business Processes
- System Modernization Strategy
Level 0 describes only the **functional business decomposition** of an enterprise in terms of high level areas of business and business/information entities.

R0 Driven by business goals

For e.g.

- We want to open up a new line of business for company insurance
- We want to have a cross functional claims process
For each high level area of business there is an L1 model which describes the lifecycle of key processes (in this case for insurance policies) that are needed for that area as well as the business entities that they require as input and emit as output.

R1 Driven by high level - Lifecycle requirements
• There must be a policy in existence prior to updating, renewing, canceling or querying a policy.
• Updating, renewing, cancelling and querying can happen at any time
• Updating, renewing, cancelling and querying can happen zero or more times until a policy is terminated
LEVEL 2: - Driven by Solution Requirements

For each lifecycle process in L1 define one or more sub choreographies that describe the dynamic behaviour of the communication model.

This model does not need to bind to an underlying concrete information model and is abstract. So no channel identities and no xpath expressions.

Level 3: Technical Requirement and Constrained by Information Model

This model does need to bind to an underlying information model and is concrete with channels having identities and conditionals with expressions.
Case Study - Level 4 and 5

Level 4
(Driven by L3 contracts)

- Generation of state machines that represents the observable behaviour of that participant.

Level 5

- Generation or Development of a fully executable application or process that retains the state behavior that is observable in L3 but adds non-observable business logic to the state machine.
Testable Architecture in Action!
[e-Claims process - movie]
In Summary

- Better “Enterprise” Architecture is achieved through:
  - Focus on components (Business, Information, Application and Technology) and their inter-relationships across the enterprise
  - Adherence to best practices for modelling to describe enterprise states

- Adoption of:
  - Formal methods of enterprise modelling to ensure consistency and predictability of outcomes
  - Testable architecture to improve architecture governance and control over implementation artefacts
  - Testable architectures as an extension to Enterprise architecture methods to help clearly articulate formal descriptions for component inter-relationships
  - Testable architecture methodology to auto-generate detailed contracts and implementation artefacts in adherence to functional and non functional requirements
Thank you

Q & A

Bhavish Kumar
Deputy Practice Leader, ASP Europe
bhavish.kumar@cognizant.com
Skype: bhavish.kumar.madurai

Further Reading:
realisticenterprisearchitecture.blogspot.com
pi4tech.blogspot.com
opengrouppresentations.blogspot.com

© 2009, Cognizant Technology Solutions. All Rights Reserved. The information contained herein is subject to change without notice