SAVARA Architecture: From Art to Engineering

Getting Serious about Enterprise Architecture

(Whitepaper on Alignment between Testable Architectures and TOGAF)

Author: Bhavish Kumar Madurai, Co-chair, SAVARA Project

Prologue

This whitepaper has been written to achieve the following 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 of Enterprise Architecture standards like The Open Group Architecture Framework (TOGAF Version 9) or Zachmann to help clearly articulate formal descriptions for the component inter-relationships
  - Benefits of Testable Architectures

Setting the Scene – Standard EA Definitions

IEEE Std 1471-2000 defines Enterprise Architecture as “the systems fundamental organisation, embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution”

TOGAF 9 defines Enterprise Architecture as

a. A formal description of a system, or a detailed plan of the system at the component level, to guide its implementation (source: ISO/IEC 42010:2007)

b. The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time.

Other major definitions detail 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 realisation of enterprise goals to enable an enterprise performance to be predictable, measurable and manageable

The key factor in the above definitions for enterprise architecture is the focus on principles, components and more importantly formal inter-relationships between components. Much of the architecture we see today do not emphasize on formal relationships between participating components which is brings the main problem of ambiguity and error within various architectural layers.

The problem domain around failed programmes and effort lost in extensive and often repeated testing lifecycles is primarily because of ambiguity in requirements (capture, analysis or engineering) and then ambiguity between architecture and requirements and finally the cascading effect of ambiguity between implementation and architecture

There is ambiguity because requirements are divorced from architecture and architecture is divorced from implementation. As architects we write a lot of documentation and create a lot of great diagrams.

However, how many of us have really proven that what we have written in terms of architecture is actually what is built finally? If proven, is the proof empirical or derived or formal?

While empirical or derived proofs (through various kinds of testing) are okay for simple projects with straight forward architectures, they do not water on large programmes and end up in extensive testing cycles which are often repeated and involve huge efforts and wastage of time.

As a result of ambiguity we end up with:

- *Poor Alignment of IT to business goals and objectives*
- *High Cost in managing complexity*
- *High cost of testing*
- *Lack of transparency and control in delivery and change management issues in large programmes*
- *Poor re-use of key IT assets*
- *Lack of Business agility hindered by inefficient IT Architectures*

So removing ambiguity by joining up things, moves us from “art to engineering” leading to the industrialisation of IT through efficient use of architecture methods

Testable architectures are the foundation of removing this ambiguity. Testable Architecture enables the architecture of a system to be described unambiguously using Choreography Description Language (CDL) such that it may be tested against requirements and is used to generate implementation artefacts for delivery thereby improving governance and control across large system integration programmes.

If we can deliver a solution that connects requirements to architecture and to implementation, we shall change the nature of complex systems delivery, reducing costs, mitigating delivery risks and improving time to market of key business functionality

Testable architecture methodology uses a unique combination of abstraction, modelling and simulation to the architecture definition process and the ordered interactions between participating components coupled with any constraints on their implementations and behaviour. Testable architecture is formal hence reduces defect injection across a programme lifecycle

Testable architecture is formally grounded and with strong type definition and has its foundations in “pi-calculus” which is a formal communication framework developed by Prof. Robin Milner – Professor Emeritus of Computer Science at the University of Cambridge and Turing award recipient:

Key benefits of Testable architectures:

- *Improved delivery assurance*
- *Reduced cost of implementation and testing*
- *Increased quality of overall solution*
- *Increased agility of overall solution*

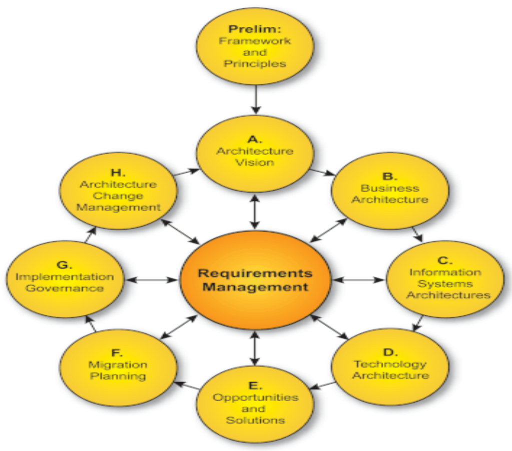
Some of the noteworthy, real life implementations using testable architectures include:

- **HL7** - Lifesciences principle messaging interchange standard (CDL provides the dynamic model for message order enabling rapid deployment of HL7 Compliant services (a.k.a. SOA)

- **ISDA –** Derivatives principle message interchange standard (CDL provides the dynamic model for confirmations, affirmations, 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

TOGAF is THE OPEN GROUP ARCHITECTURE FRAMEWORK which is the collective effort of many organisations (consulting, system integrators, and end users) within the architecture forum and it details processes, methodology and artefacts for efficient and effective delivery of enterprise architectures for any organisation regardless of the size (i.e. being scale invariant)


Prelim: Framework and Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

TOGAF 9 stands out as an important and well accepted standard for Enterprise Architecture with key artefacts, methods and processes to detail architecture of any size or complexity. These components and methods include:

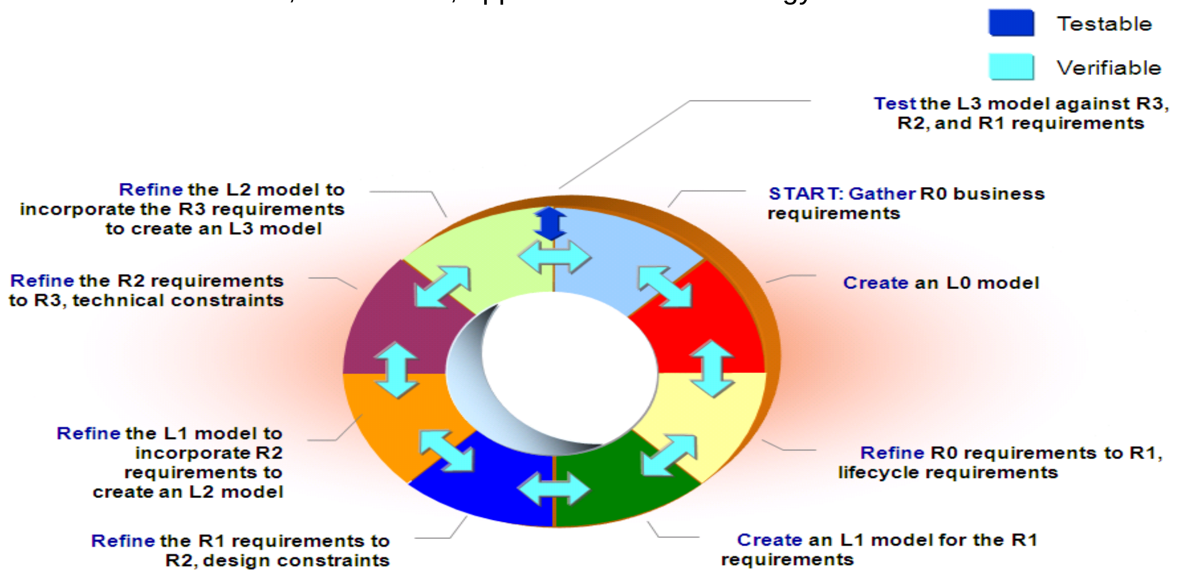
- *The main iterative crop circles framework to define architecture*
- *The content framework defining a clear standard for architecture documentation*
- *Reference Models for business, information and technology architecture*
- *Enterprise Continuum (contains the Architecture continuum and Solutions Continuum)*
- *Architecture Capability Framework (to help organisations build an architecture organisation)*

The benefits using TOGAF's Architecture Definition Method 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 artefacts and methods etc..)
- **Efficiency**
  - Creates a repeatable and predictable process of developing enterprise architecture content
  - TOGAF ADM can be extended and customised as per the specific needs of the enterprise for e.g. scaling
- **Simplicity**
  - TOGAF ADM is Process Driven : Inputs, Outputs and Steps are specified for each phase of the iterative framework
- **Predictability of Outcome**
  - The outputs from one phase could be tracked back to the inputs for the next phase – i.e TOGAF ADM links inputs to Outcomes

Testable Architecture Methodology

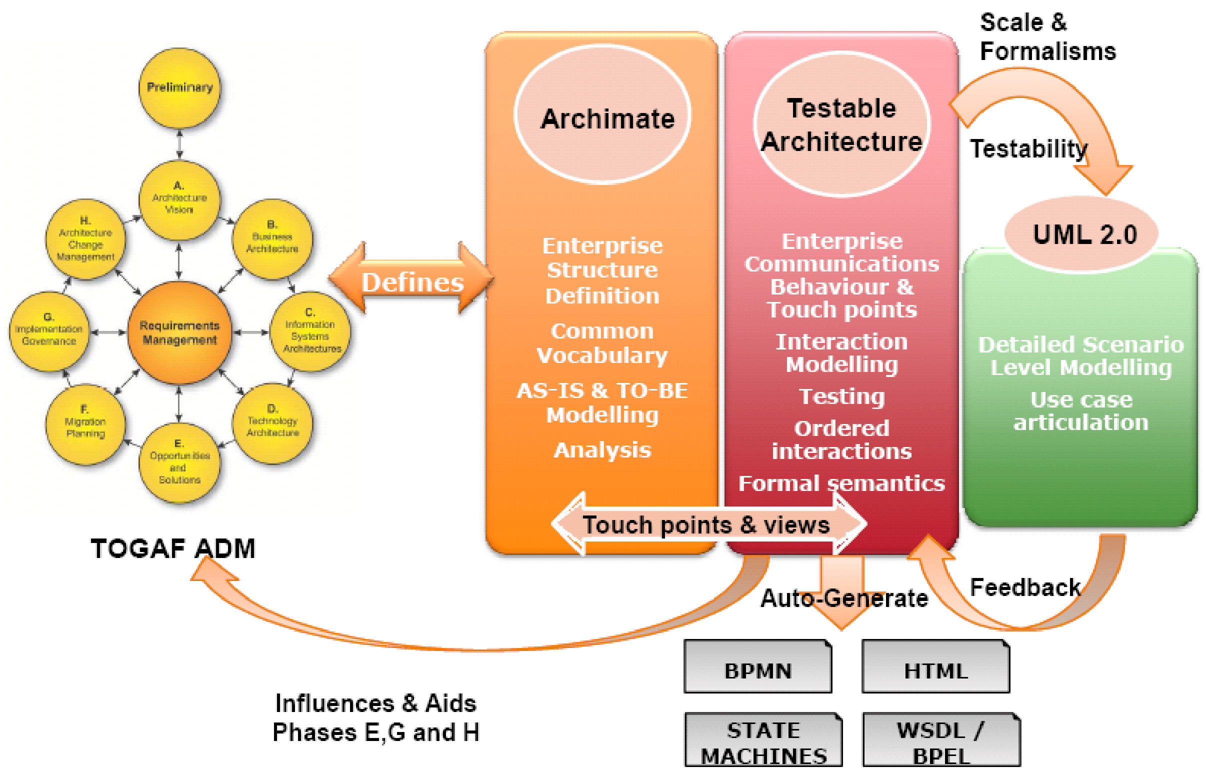
The diagram below gives a brief overview of the testable architecture methodology which complements the TOGAF Iterative methodology given in the previous section across various architecture views business, information, application and technology architectures


Testable
Verifiable
Test the L3 model against R3, R2, and R1 requirements
START: Gather R0 business requirements
Create an L0 model
Refine R0 requirements to R1, lifecycle requirements
Create an L1 model for the R1 requirements
Refine the R1 requirements to R2, design constraints
Refine the L1 model to incorporate R2 requirements to create an L2 model
Refine the R2 requirements to R3, technical constraints
Refine the L2 model to incorporate the R3 requirements to create an L3 model

Successive refinement of requirements and models leading to streamlined delivery

Alignment of Modelling Methods

Modelling languages like Archimate help alleviate the issues around ambiguity by defining enterprise structure. However testable architecture aligns to TOGAF by describing the enterprise communications behaviour. Testable architecture adds scale and formalisms to UML and auto generates implementation artefacts that help in removal of ambiguity and thereby deliver robust solutions. The diagram below shows a pictorial representation of the alignment between these methods.



Testable Architecture and link to SAVARA

SAVARA, is the next generation of Testable Architecture' methodology, that aims to give software architects insight into IT implementations at the architecture and design stage, meaning business scenarios can be modelled and changes made much earlier in the typical software testing cycle - before a single line of code has been written. Empirical research from Roger. S. Pressman(an internationally recognized consultant and author in software engineering) cost of correcting design defects at the traditional testing stage to be around 200% more than correcting them in during requirements or architecture stage. This is similar to research published by SEI Capability Maturity Model. SAVARA aims to dramatically reduce testing expenditure and overall software development costs through modelling and simulation and makes it enterprise scale.

With development budgets getting tighter and the need for agility becoming more important, there is simply no need for architectural errors to still be present in the testing stage of IT projects. They're expensive and time consuming to fix and, crucial business requirements fall through the gaps. By bringing in a high level of testing rigour, measurement and formalism to SOA and the software development lifecycle, SAVARA will deliver real returns for customers, reducing the cost of ongoing projects, and freeing up budget for further, revenue-generating initiatives

Testable Architecture' the foundation of SAVARA ensures that artefacts defined in each phase of the software development lifecycle (e.g. business requirements, architectural models, service designs, code, etc.) can be verified for conformance. For example, architectural models can be verified against requirements, service designs against architectural models and code against service designs. This guarantees that the deployed systems can be shown to implement the originating business requirements.

Epilogue

Better Enterprise Architectures are 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 and communications behaviour
- Adoption of formal methods for enterprise modelling like Testable Architecture (CDL) to ensure consistency and improve predictability of outcomes
- Adoption of testable architectures to improve architecture governance and control over implementation artefacts
- Usage of Testable architecture as an extension to Enterprise architecture methods to help clearly articulate formal descriptions for component inter-relationships
- Usage of Testable Architecture methodology to auto-generate detailed contracts and implementation artefacts in adherence to functional and non functional system requirements

Further Reading

For further information please visit
http://www.jboss.org/savara
http://realisticenterprisearchitecture.blogspot.com/
http://pi4tech.blogspot.com/