Model Based Test Architecture: An Abstraction to Improve Reality

Raytheon Missile Systems
George M Hollenbeck
Javier O Villafañe
October 3-6, 2016
Agenda

- Introduction
- Test Architecture Overview
- Test Architecture from Engineers Point of View
- Test Architecture Using Model Based Elements
- Knowledge Points and Development Process
- Test Architecture Implementation
- Conclusion
Introduction

- **Test Architects** – chief builder of test
- **Test Architecture (TA)** – the explicit definition of a test program that Test Architects defines the structure in terms of components, connections, and constraints of a product, process or element
- **Model Based Test Architecture (MBTA)** – the explicit definition of a model that supports the Test Architect to define the Test Architecture

---

TA from Engineers POV

Low fidelity TA model
Systems Domain & Test Domain are concurrent and coupled with Test lagging

Medium fidelity TA Model
Systems Domain drives Design
Design drives Test Domain
Communication is key for success, modeling increases cross discipline communications & artifacts

High fidelity TA Model
Collecting data during IV&V

Deploy & Maintain

Test & Eval

Modeling fidelity increases as the program matures
TA from Engineers POV

- Systems Engineering & Test Architects working on modeling the Unit Under Test (UUT) and the Test Architecture (TA) concurrently will provide higher understanding with less defects by increasing multi-disciplined communication using a industry support language SysML and its associated tools.

- Model Based Test Architecture (MBTA) enables managing data, generating artifacts and having information available to the entire team which is easier than traditional methods (distributed documents and sources).

- Architecture leads into design and design implementation becomes a UUT in its test environments (TEn).

As the slide rule is to the calculator the MBTA will be to test...
Test Architecture Overview

Foundation of the Model Based Test Architecture (MBTA)
Knowledge Points (KP)

Captures what activities or what we need to know about the system that allows it to meet the system intended purpose

SysML Element

MBTS

Inputs

Outputs

Knowledge Point Ontology

TA

SME

Sys Rqmt

Risk

Operational UC

Maintaining the focus on what is required!
Test Events

Primary purpose is to satisfy a KPs and to verify requirements when the data is produced and analyzed.

Test Events Ontology
Test Architecture

Captures approach to obtain objective evidence!
Test Environment (TEn)

Provides the capabilities awareness that enable Test Engineers to determine which assets and artifacts are required to execute a test event.

- Test Environment Ontology
- MBTS
- Test Environment
- Test Capability
- Risk
- Test Data
- Schedule
- M&S
- TEn
- Facilities
- Test Team

Captures the Test configuration and Information flow.

Example for reference

Modeling the details of Test Environment
Test Strategy

Test strategy is capable of evolving as the system progresses throughout its lifecycle.

SysML Activity Diagram

Example for reference

Modeling and generating a test strategy
Test Architecture Modeling

SysML Test Architecture Model

Example for reference

Test Architecture Taxonomy from the Test Event perspective
Test Architecture perspectives

Facilities to Events

Events to TEn

Req’s to Events

God’s Eye View of the TA

Tool generated artifacts and views
Knowledge Points and Dev Process

Establishes the framework that enables the creation of actionable events and assist in the overall synchronization of the test effort towards a common goal.

KPs work with different workflow methods.

Sources of Knowledge:
- Subject Matter Experts
- Risk
- System Requirements

KP Identification
KP Definition
KP Generation

Agile Methodology
Source: http://www.scaledagileframework.com/

Waterfall Methodology
Source: http://www.umsl.edu/~hugheyd/is6840/waterfall.html
Test Architecture Implementation

- There is a cost of creating a Test Architecture from the start of a program and maintaining that model through the lifecycle.
- The infrastructure and work processes will be different with the net results also being improved quality and execution of a product satisfying the customers needs which should justify the return on the investment (ROI).
- Increases repeatability of the work flow processes via the use of a model and a defined ontology.
- The test architecture will provide the entire team increased knowledge of the test strategy and test details using a modeling tool.

ROI is justified by increased quality and effective test results.
Conclusion

- Increased cross-disciplined communications equates to decreased defects and reduces overall costs and schedule complications.

- Modeling using the language of test with the same language modeling the design enables interoperability modeling.

- Defining the test language and defining the test architecture is critical for ensuring a test strategy is understood and available.

- Modeling the test program supports traditional and alternative work flow processes such as Agile and waterfall methods.
Questions?