Model-Centric Systems Engineering (MCSE) in an Agile Environment

Natasha Shevchenko

Mary Popeck

Tim Morrow

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM20-0966

MCSE in an Agile Environment

Introduction

Why Complex Systems Require Systems Engineering/ Architecture Management

How to Integrate MCSE with Agile Practices

Conclusion

MCSE in an Agile Environment

Introduction

The Agile Manifesto

Agile uncovers better ways of developing software by valuing:

- ***Individuals and Interactions*** over Processes and Tools
- ***Working Software*** over Comprehensive Documentation
- ***Customer Collaboration*** over Contract Negotiation
- ***Responding to Change*** over Following a Plan

While there is Value in the Items on the Right, the Items on the *Left* are Valued More.

For Systems Engineering, moving to the left means earlier identification of risks, earlier refinement of the system vision and its requirements, and earlier identification of testing needs.

Systems Engineering

Increasing complexity of systems leads to increases in system risk and unreliability.

Complex systems need to be engineered → System Engineers design, integrate and manage complex systems over their life cycle

Examples of tools and methods used by System Engineers to comprehend and manage complexity include:

- System Architecture
- System Modeling & Simulation
- System Analysis

Systems Engineering and System Architecture are tightly connected.

System Architecture at a High Level

- A set of structures needed to reason about a system, the relations among those structures, and the properties of both the structures and the system.
- An Architecture View is a representation of one or more structures, as written by and read by system stakeholders.
- Structures represent the primary engineering leverage points of an architecture and allow for the manipulation/trade-off of quality attributes/non-functional requirements (e.g., Performance, Security, Reliability).
- Every system has an architecture whether it is documented or not.
- There is no such thing as a good or bad architecture; Architectures are either more or less Fit for some Purpose.

The Architecture conveys a Vision of the System along with a way to achieve it!

Two Different Approaches

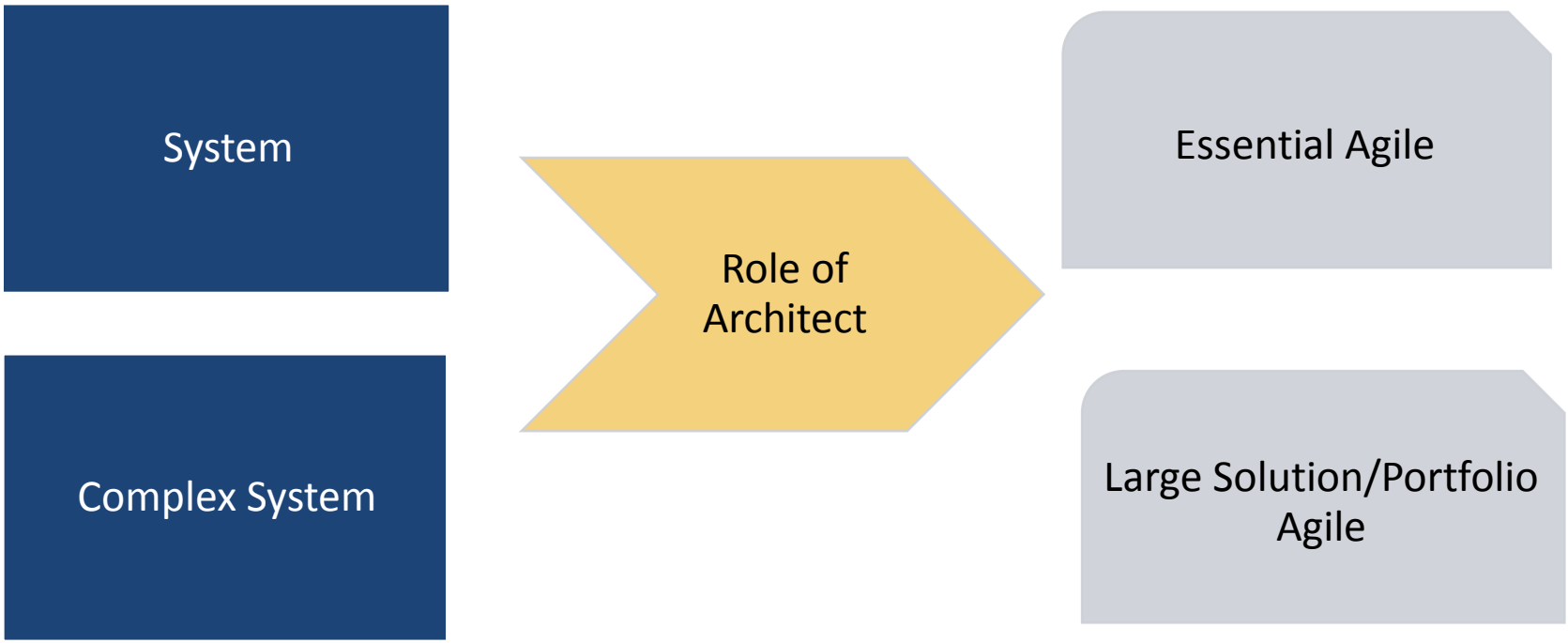
Agile Practices	Architecture Practices
Frequent requirement changes are embraced	Requirements and quality attributes are identified early
Early and continuous delivery with short delivery cycles	Significant upfront investment before any delivery
Short planning horizon (near-term focus)	Provides strategic (big-picture) view of the system (long-term focus)
Requirements and architecture emerge	Solution alternatives are analyzed and simulated before any development. (Requirements and architecture are planned and well documented.)
…	…

Tension Between Agile and System Architecture

- Current agile practices work well for systems that incorporate well-established architectural patterns where a pattern is a known solution to a common problem in a given context.
- For complex systems with custom patterns, the sustainability of the system will be adversely impacted if the architectural decisions are not adequately communicated to the agile teams and properly implemented.
- Requirements and required capabilities such as security & modifiability need to be engineered into the architecture before they are passed to the agile teams.
- The architecture changes of complex systems must be managed as they may impact the entire system forcing extensive major revisions.

The System Engineer must address these tensions and facilitate performance of the necessary system trades!

Scaled Agile* Requires Architecture Management



* Scaled Agile means to scale lean and agile practices beyond a single team.

Why Complex Systems Require Systems Engineering

What Scaled Agile is Still Missing

- Identification of architecture and business drivers
- Identification of system quality attributes/non-functional requirements such as security, performance & modifiability
- Identification of assumptions and constraints
- Ability to make, communicate and document design decisions and trade offs
- Ability to analyze the architecture to identify impact of changes and possible risks early in development lifecycle

The role of the architect must be emphasized more!

Can Architecture be Agile?

- Dynamic and Up-to-date Documentation
- Changes made and communicated quickly to all stakeholders and teams
- System analysis/impact analysis performed efficiently
- Architecture processes as agile as the other development processes

How to do architecture practices in an agile fashion?

What is Model-Centric Systems Engineering?

- Not yesterday's Document-Centric Systems Engineering
- Systems Engineering uses a Digital System Model* to facilitate common system understanding and decision-making.
- The Digital System Model* is the single authoritative source of truth
- System and Components can be integrated at various levels of abstraction and fidelity
- Model Views are chosen to best communicate information to a variety of stakeholders via the dynamic creation of multiple, consistent, accurate views
- Impacts of changes are more easily analyzed and evaluated

Digital Modeling allows architecture to be more agile!

*Note: The Digital System Model contains the most current requirements, key mission/business operations, architecture, design details, implementation details, test and evaluation details, and supporting documentation.

Scenarios

Scenario 1: Non-Functional Requirements Traceability

- Complex System with dozens of capabilities and hundreds of requirements
- New Capability is introduced
- How to do Requirements traceability including Non-functional requirements (i.e., security, performance, etc.) with and without MCSE?

Scenario 2: Impact Analysis

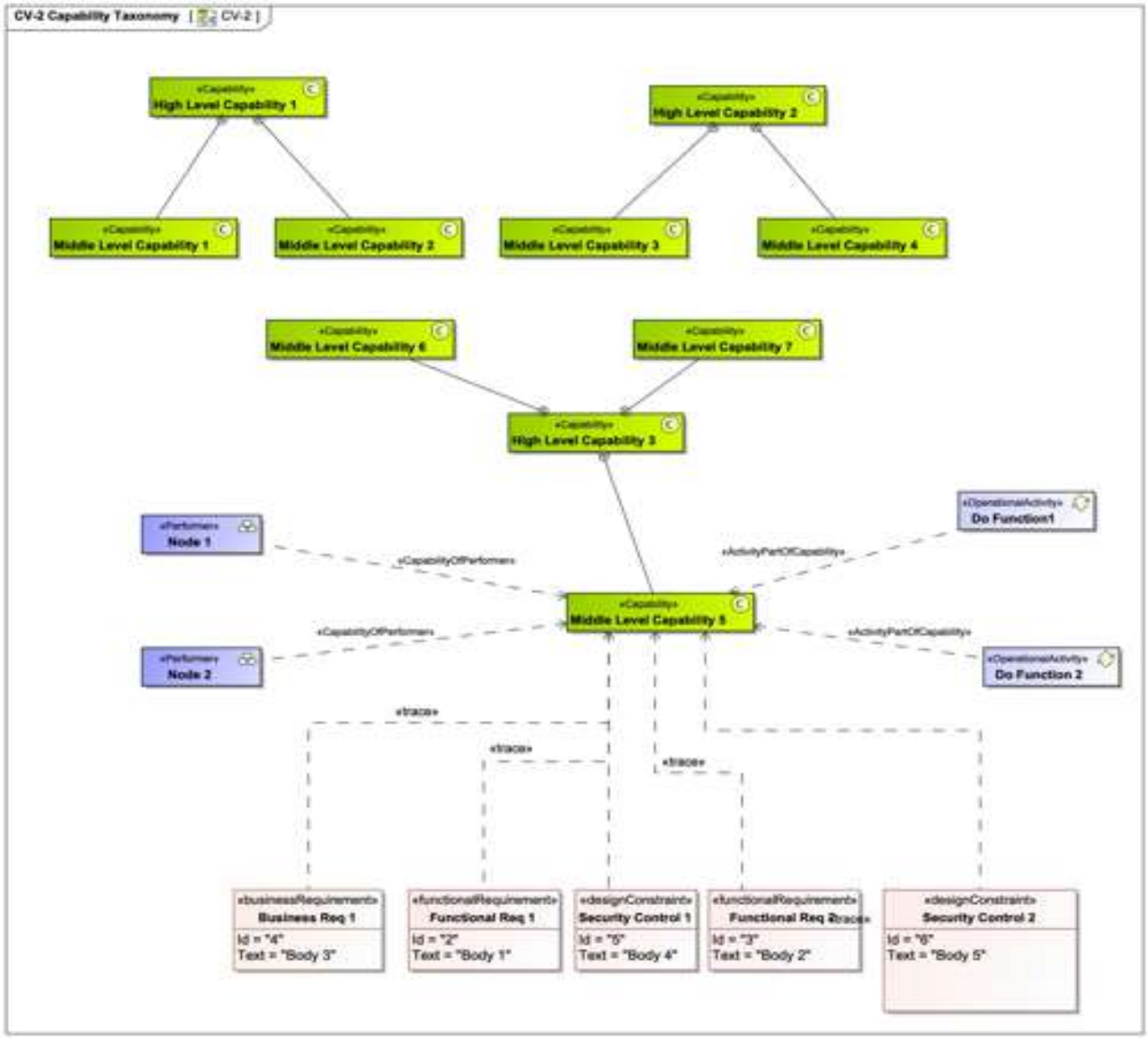
- Complex System with dozens of capabilities implemented by several sub-systems
- Existing Capability needs a significant change
- How to do Impact Analysis with and without MCSE?

Scenario 1: Non-Functional Requirements Traceability

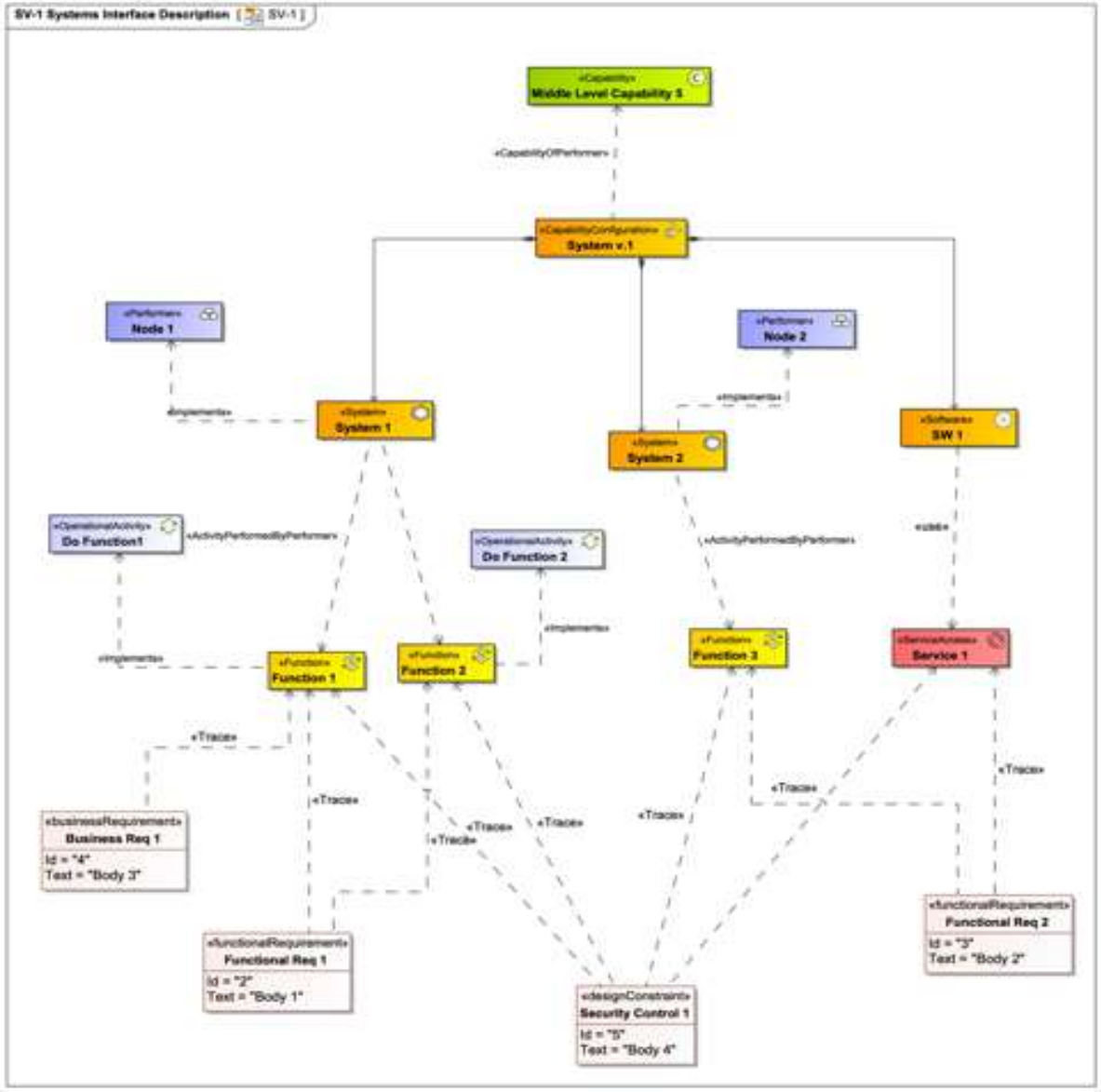
1. New Capability is requested
2. Set of requirements are developed, including non-functional ones
3. Capability is decomposed, and Business Activities are designed
4. Solution Architecture is developed, architectural alternatives are considered including technology choices
5. System Architecture is built together with platform solution
6. Implementation roadmap of the solution is decomposed to Features and Stories

For Document-Centric Systems Engineering, what does it take to ensure that non-functional requirements are designed-in and implemented?

Scenario 1: MCSE – Capabilities Decomposition

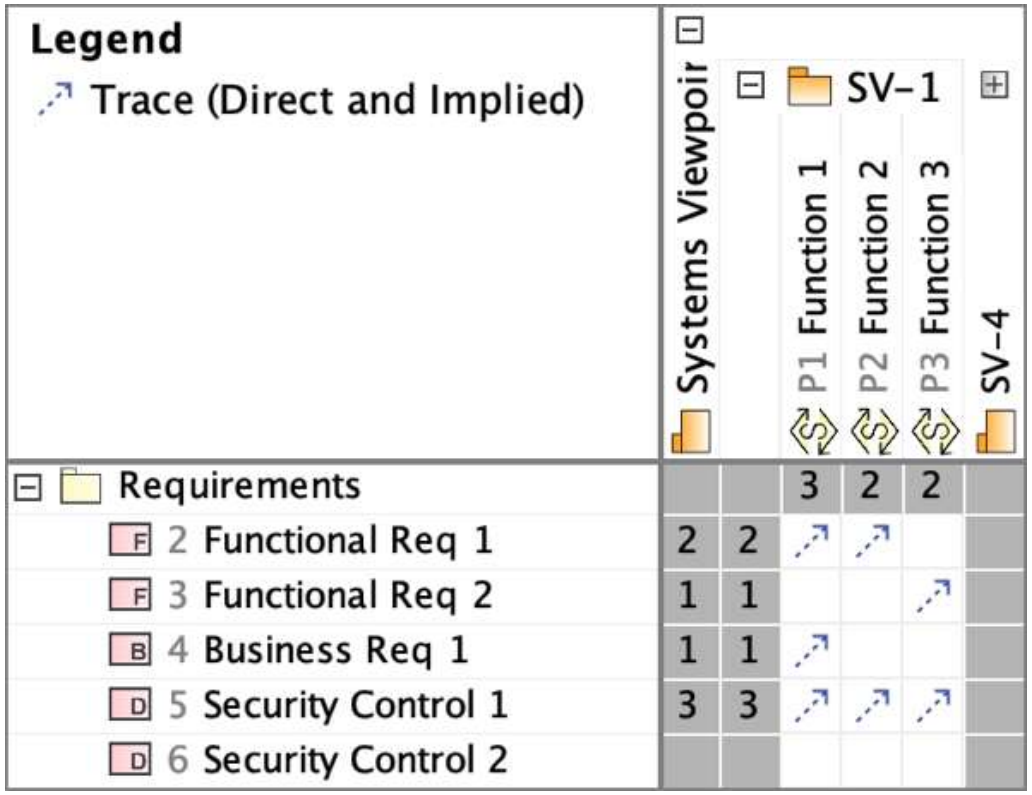


Scenario 1: MCSE – System



Scenario 1: MCSE - Traceability Matrix

Traceability Matrix supported by Digital Modeling Tool allows to dynamically identify “missed” Requirements.



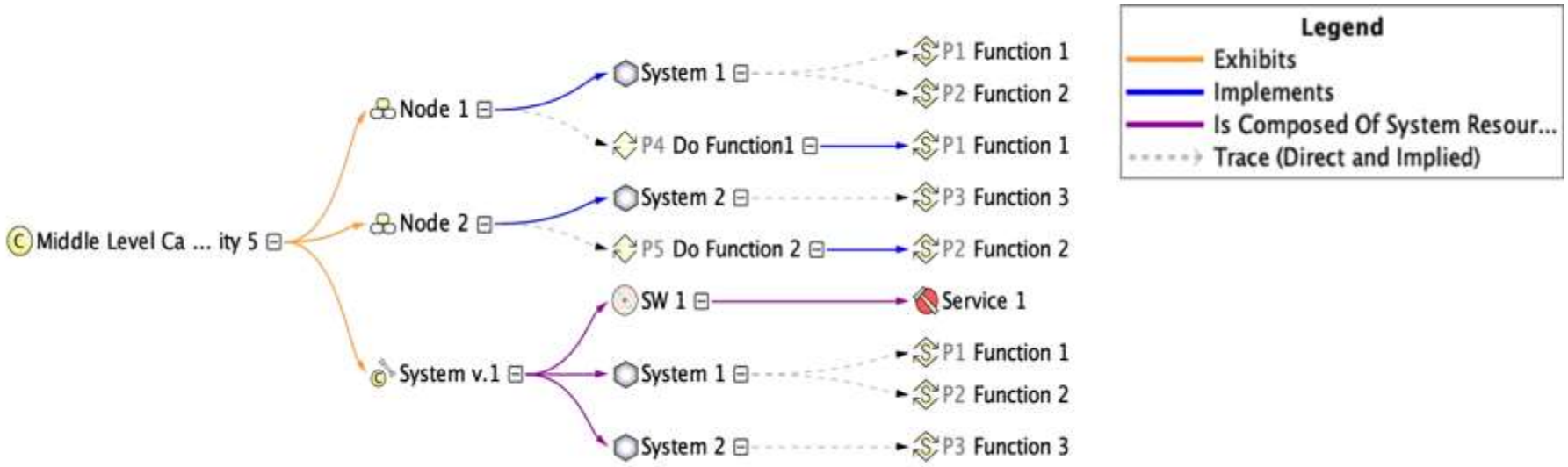
Scenario 2: Impact Analysis

1. Significant change to one of the Capabilities is requested
2. Set of requirements are developed
3. Changes to the Capability are decomposed, and new Business Activities are designed

For Document-Centric Systems Engineering, what does it take to do an impact analysis?

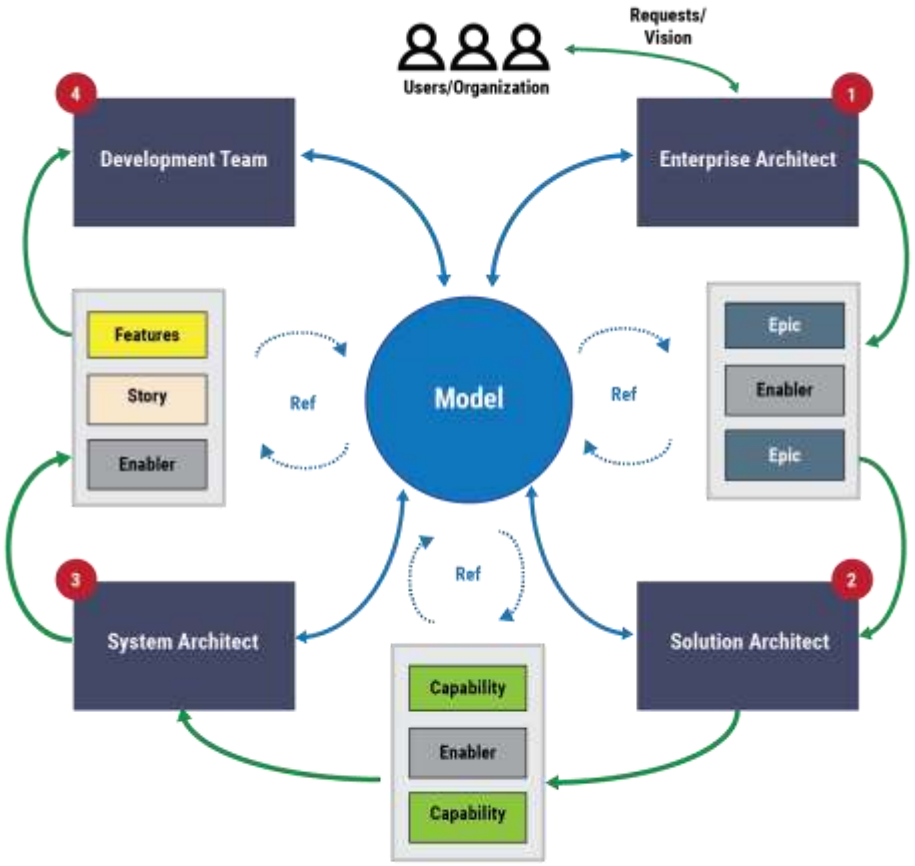
Scenario 2: MCSE - Implementation Map

Implementation Map supported by Digital Modeling Tool allows to dynamically identify impacted by the changes resources.



How to Integrate MCSE with Agile Practices

Proposed Interaction of System Engineering/Architecture and Agile Roles



Marrying Architecture Practices with Agile Practices is Possible!

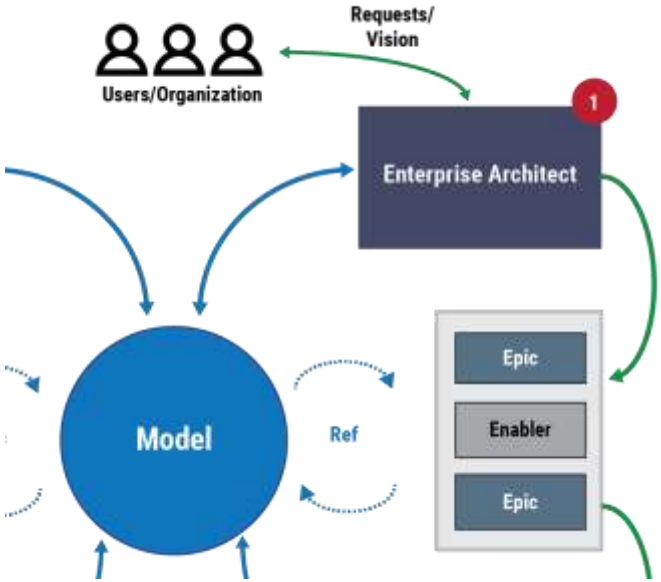
Complex Systems Architecture Mapped to Agile

Agile Architectural Role: Enterprise Architect

Activities:

- Work on Enterprise Architecture ***in the Model***
- Identify Business Epics and Enabler Epics ***using the Model***
- Decompose Business Epics and Enabler Epics to Capabilities, Features and Enablers for Value Streams and Systems ***using the Model***
- Reference ***Model*** Elements in corresponding Tickets
- Reference Tickets in corresponding ***Model*** Elements

Keep the Model Current!



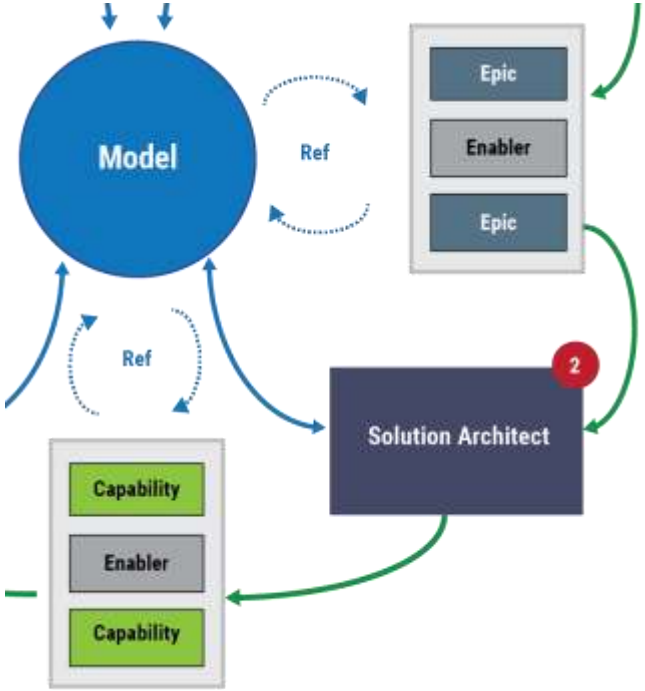
Large Systems Architecture Mapped to Agile

Agile Architectural Role: Solution Architect

Activities:

- Work on Large System Solution Architecture ***in the Model***
- Identify Technical Capabilities and Enablers ***using the Model***
- Decompose Technical Capabilities into Features and Enablers for Systems ***using the Model***
- Reference ***Model Elements*** in corresponding Tickets
- Reference Tickets in corresponding ***Model Elements***

Keep the Model Current!

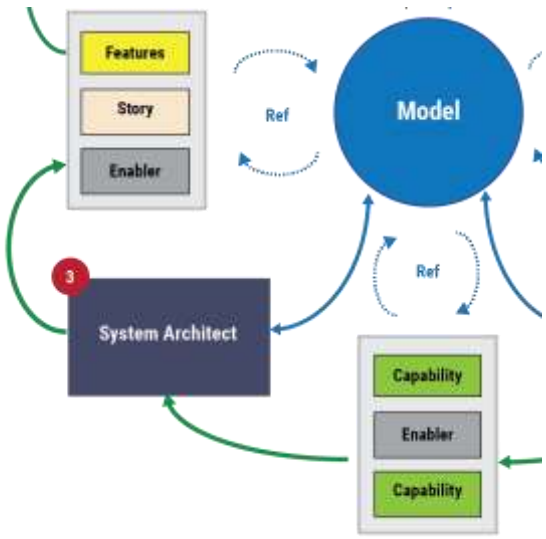


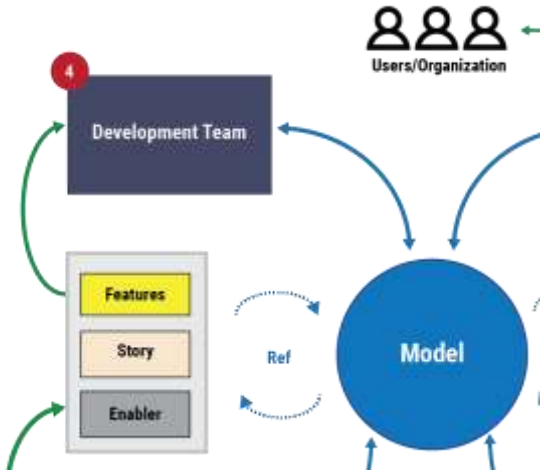
System Architecture Mapped to Agile

Agile Architectural Role: System Architect

Activities:

- Work on System Solution Architecture ***in the Model***
- Identify Features and Enablers ***using the Model***
- Decompose Features and Enablers in to Stories ***using the Model***
- Reference ***Model Elements*** in corresponding Tickets
- Reference Tickets in corresponding ***Model Elements***





Keep the Model Current!

MCSE in an Agile Environment

Conclusion

In Conclusion,

- MCSE integration with agile practices improves complex system implementation by facilitating the execution of trades and analyses early in the process, enabling multiple development teams to work together more efficiently and effectively.
- MCSE produces a system vision with contextual information not normally available when following agile practices. Yet, it is the glue that holds a complex system together.
- Integrating Ticket Management applications into MCSE will further the adoption of MCSE into agile practices, and should be encouraged.

Merging MCSE and Agile Practices will improve Quality, Schedule and Cost Performance for Complex Systems!

MCSE in an Agile Environment

Backup

Architecture Framework

An architecture framework is a partial design that provides services which are common in particular domains.

It constrains the initial design hypothesis.

It divides the architecture description into domains, layers, or views and offers models for documenting each view.

It allows for making systemic design decisions on all system components and making long-term decisions.

A framework is **NOT**:

- An Architecture
- A Methodology to develop Architectures

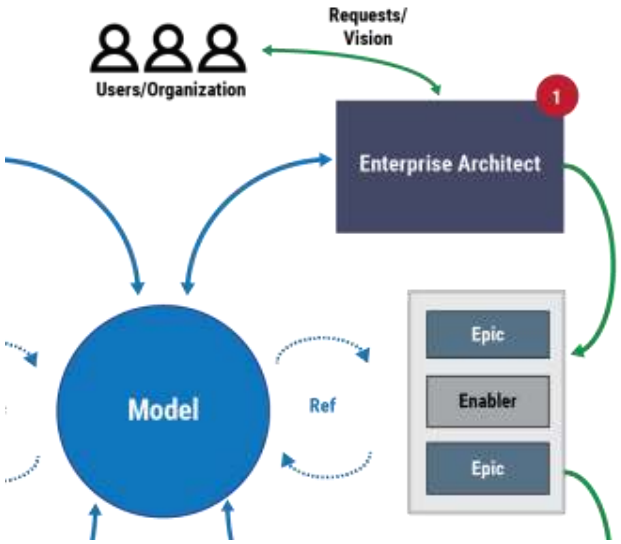
Framework examples include: TOGAF, DoDAF, MODAF, ESAAF, FEAF and UAF.

MCSE Checklist

- Ensure MCSE tools include a Ticket Management System and an MBSE Digital Modeling Environment (DME)
- Ensure MCSE tools are set up and configured before project start
- Ensure all team members can access the necessary tools including the Architect, the Test Engineer and don't forget the Technical Writer
- Conduct training on MCSE tools
- Establish business processes to ensure all architectural and design work is done in the DME (i.e., the model)

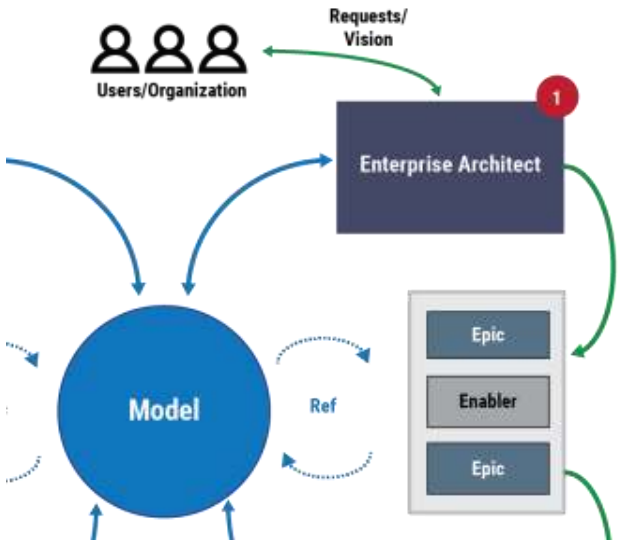
The System Vision and Strategy

- System Concept and Goals
  - OV-1 view
  - CV-1 view
- Requirements
  - Requirements Diagrams
- High-level Capability Taxonomy
  - CV-2 view



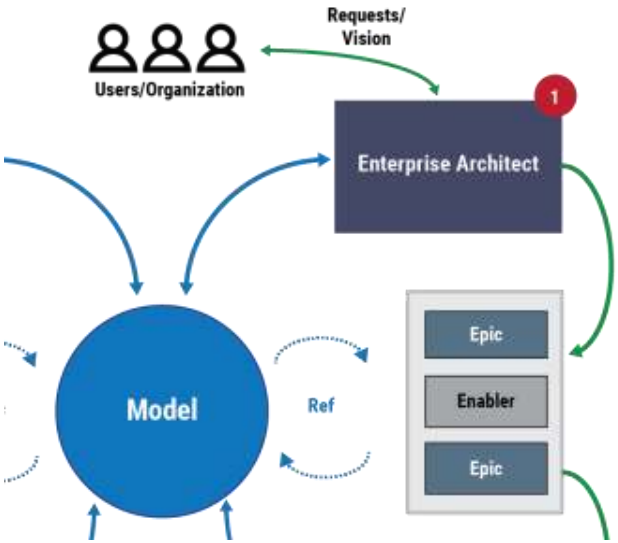
The System Definition

- External Interfaces:
  - OV-1 view
- Value Chains
  - Use Cases Flows
  - OV-5b view
- Quality Attributes
  - Requirements
- Standards
  - Standards viewpoint



Building the Roadmap

- Business Capability Decomposition
  - CV-2 view
- Capability Dependencies
  - CV-4 view
- Capabilities Phasing
  - CV-2 view
- Performers and Information Flows
  - OV-2 view
- Business Activity Flows
  - OV-5b view



Building Portfolio Backlog

Identify Business Epics and Enablers from

- High-level Business Capabilities
- Operational Activities
- Capability Enablers

Include References to the model:

- Put a reference (URL) to the Business Capability's element in the model in the corresponding tickets
- Put a reference to the ticket in the Business Capability documentation

Building Solution and Program Backlog

Identify Technical Capabilities, Features and Enablers from

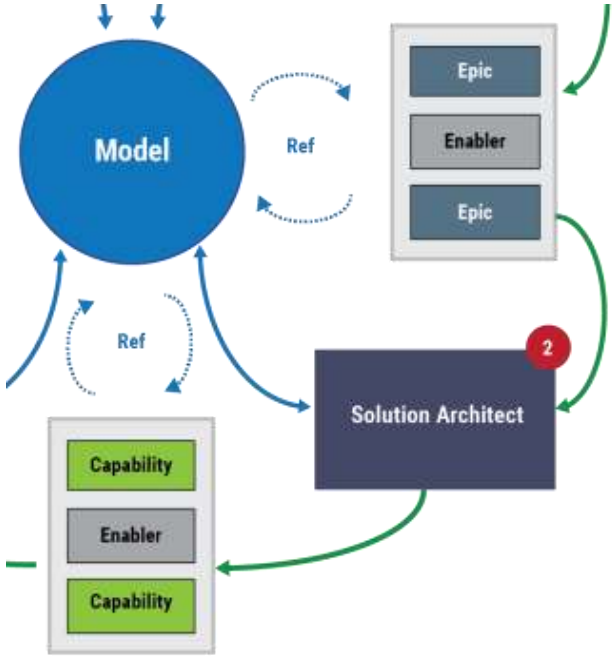
- Lower-level Business Capabilities
- Operational Activities
- Capability Enablers

Include References to the model:

- Put a reference (URL) to the Business Capability's element in the model in the corresponding tickets
- Put a reference to the ticket in the Business Capability documentation

First Cut of the System Architecture

- Components and Interfaces
  - SV-1 view
  - SV-2 view
  - Mapping to Capabilities
  - Services allocation
- Scenarios Flows and States
  - SV-4 view
  - Svc-4 view
- Services Definition
  - Svc-1 view
  - Svc-2 view
  - Mapping to Capabilities



Generic

JIRA	Agile	Architecture Model
Initiative	Epic	Capability
Epic	Capability	Capability Configuration
Feature	Feature	Service/Function
User Story	Story	

Portfolio + Essential

Environment	Term	Definition
Architecture Model	Capability	A Capability is the ability to achieve a desired effect under specified [performance] standards and conditions through combinations of ways and means [activities and resources] to perform a set of activities.
	Capability Configuration	A Capability Configuration is a composite structure representing the physical and human resouces (and their interactions) in an enterprise, assembled to meet a capability.
	Service/Function	A Function is an activity which is specified in the context of the Resource Performer (human or machine) that is capabileof performing it.
Agile	Epic	An Epic is a container for a significant Solution development initiative that captures the more substantial investments that occur within a portfolio.
	Capability	A Capability is a higher-level solution behavior that typically spans multiple ARTs. Capabilities are sized and split into multiple features to facilitate their implementation in a single PI.
	Feature	A Feature is a service that fulfills a stakeholder need.
	Story	A Store is a short description of a small piece of desired functionality, written in the user's language.
	Enabler	An Enabler supports the activities needed to extend the Architectural Runway to provide future business functionality.
Jira	Initiative	A Initiative are collection of epics that drive toward a common goal.
	Epic	An Epic is a large body of work that can be broken down into a number of smaller stories, or sometimes called "Issues" in Jira. Epics often encompass multiple teams, on multiple projects, and can even be tracked on multiple boards.
	Feature (custom)	A Feature is smaller epic (see Agile definition).
	User Story	Stories, also called "user stories," are short requirements or requests written from the perspective of an end user.