INTERFACE MANAGEMENT WITH MBSE – FROM THEORY TO MODELING

Matthew Hause
Engineering Fellow, MBSE Specialist
October, 2017
AGENDA

1. Introduction
2. Interfaces
3. System of System Interfaces
4. System Interfaces
5. Through the development lifecycle
6. Conclusion
INTRODUCTION

• Interoperability is a key facet of a successful system, and essential to a system of systems.

• Interoperability is a property of a system, whose interfaces are completely understood, to work with other products or systems without any restricted access or implementation.

• Software interoperability is the capability of different programs to exchange data via a common set of exchange formats, (read/write) file formats using same protocols.

• DOD: The condition achieved among communications-electronics systems when information or services can be exchanged directly and satisfactorily.

• So, interoperability begins with interfaces: mechanical, electronic, hardware, software, people-ware, etc.
DESIGNING INTERFACES

• Starts with requirements and stakeholder needs

• System-to-System interfaces
  – Define the required behavior/functionality
  – Identify the Dependencies - interaction with other systems and within the subsystems
  – Identify the necessary interactions
    • Data, physical, logical, electrical, etc.
  – Define logical interface requirements
  – Define interaction performance characteristics
  – Allocate to physical interfaces

• Human Interfaces
  – Identify the characteristics of the (Human) users that will interact with the system.
  – Define the required tasks to be performed
  – Identify the Primary User Interface Elements
  – Define the Navigation Map
FULL PORT NOTATION

Full Port (directional notation derived)

Part1
«full»
FullP1 : IB1

Interface Block
«interfaceBlock»
IB1
flowProperties
«FlowProperty» out : sig1

Part2
«full»
FullP2 : IB2

«itemFlow»
Item Flow1 : sig1

«signal»
sig1

© 2017 PTC Permission granted to NDIA to publish and use.
SYSTEMS OF SYSTEMS INTERFACES
OPERATIONAL CONCEPT GRAPHIC

Provides a means to communicate with non-technical stakeholders while maintaining model consistency.

Defines nominal interfaces between conceptual entities in the context.

Replaced boxes with graphics.
Capability dependencies provide context for capability phases and resource deployment.

Dependencies between capabilities implies interfaces between implementing systems.
LOGICAL ARCHITECTURE INTERACTIONS

Interactions crossing swimlanes defines system interface characteristics
### Logical Architecture ICD (Fragment)

**Architectural Description** Structure [OV-3 Info Exchange]

<table>
<thead>
<tr>
<th>Operational Name</th>
<th>Conveyed</th>
<th>Producer (Operational)</th>
<th>Activity (Operational)</th>
<th>Needline</th>
<th>Consumer (Operational)</th>
<th>Activity (Operational)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHQ-BC-EL</td>
<td>Exchange Elements</td>
<td>Corporate HQ</td>
<td>Bill</td>
<td></td>
<td>BC - CHQ</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>BC-VEH.PK</td>
<td>System</td>
<td>Business Customer</td>
<td>Parcel</td>
<td></td>
<td>BC - VEH</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>VEH-BC.PK</td>
<td>System</td>
<td>Vehicle</td>
<td>Parcel</td>
<td></td>
<td>BC - VEH</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>SF-DC.PK</td>
<td>System</td>
<td>Stoveforth</td>
<td>Parcel</td>
<td></td>
<td>SF - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>DC-VEH.MN</td>
<td>Exchange Elements</td>
<td>Distribution Center</td>
<td>Manifest</td>
<td></td>
<td>VEH - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>DC-VEH.PK</td>
<td>System</td>
<td>Distribution Center</td>
<td>Parcel</td>
<td></td>
<td>VEH - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>DC-VEH.PW</td>
<td>Exchange Elements</td>
<td>Distribution Center</td>
<td>Parcel Waybill</td>
<td></td>
<td>VEH - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>DC-VEH.RT</td>
<td>System</td>
<td>Route</td>
<td></td>
<td></td>
<td>VEH - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>VEH-DC.MN</td>
<td>Exchange Elements</td>
<td>Vehicle</td>
<td>Manifest</td>
<td></td>
<td>VEH - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>VEH-DC.PK</td>
<td>System</td>
<td>Vehicle</td>
<td>Parcel</td>
<td></td>
<td>VEH - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>VEH-DC.SP</td>
<td>Exchange Elements</td>
<td>Vehicle</td>
<td>Route Status</td>
<td></td>
<td>VEH - DC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>HC-VEH.PK</td>
<td>System</td>
<td>Home Customer</td>
<td>Parcel</td>
<td></td>
<td>VEH - HC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>HC-VEH.PW</td>
<td>Exchange Elements</td>
<td>Home Customer</td>
<td>Parcel Waybill</td>
<td></td>
<td>VEH - HC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>HC-VEH.PY</td>
<td>System</td>
<td>Payment</td>
<td>Home Customer</td>
<td></td>
<td>VEH - HC</td>
<td>Performer (Operational)</td>
</tr>
<tr>
<td>VEH-HC.PK</td>
<td>System</td>
<td>Home Customer</td>
<td>Parcel</td>
<td></td>
<td>VEH - HC</td>
<td>Performer (Operational)</td>
</tr>
</tbody>
</table>

Generated automatically from the architecture.
SYSTEM INTERCHANGE SPECIFICATION

Owning Context
System
Interface
Exchange

Systems can also be specified as services
Defines system and human interface requirements and interactions

© 2017 PTC Permission granted to NDIA to publish and use.
THE EVOLUTION OF STANDARDS OVER TIME

<table>
<thead>
<tr>
<th>Standard</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 13407</td>
<td>T251</td>
</tr>
<tr>
<td>ISO/IEC 8802-3:1996</td>
<td></td>
</tr>
<tr>
<td>IETF RFC 2058</td>
<td>IETF-secash-architecture-05</td>
</tr>
<tr>
<td>IETF RFC 1828</td>
<td>IETF RFC 1510</td>
</tr>
</tbody>
</table>

Defines standards and standards forecasts.
SYSTEM INTERFACE SPECIFICATION

Defines how systems will interact to provide capabilities
## Standards Compliance Matrix

### [Architectural Description] Technical Views [StdV-1 Matrix]

<table>
<thead>
<tr>
<th>Conforming Elements</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceRole BK</td>
<td>X</td>
</tr>
<tr>
<td>ResourceRole MH</td>
<td>X</td>
</tr>
<tr>
<td>SystemInterface MH-BK</td>
<td>X</td>
</tr>
<tr>
<td>Performer(System)WN</td>
<td>X</td>
</tr>
<tr>
<td>ResourceRole WN</td>
<td>X</td>
</tr>
<tr>
<td>ResourceRole WP</td>
<td>X</td>
</tr>
</tbody>
</table>

**Generated automatically. Summarizes standards conformance**
DRIVER-HANDHELD MODULAR INTERFACES
• The order and timing of the interactions is just as critical as the interface definition itself: not just what happens, but when and why it happens.
DERIVING SERVICES FROM CAPABILITIES

© 2017 PTC Permission granted to NDIA to publish and use.
SYSTEMS INTERFACES
USE CASES DEFINE INTERACTIONS WITH ACTORS
LOGICAL V. PHYSICAL MODELING WITH IBDS

• IBDs can be used to capture both a logical model of parts, connections and flows, and a physical model

• Logical model focuses on logical parts and flows and may not show ports or types (unless logical types defined)
  – Based on specification rather than implementation (‘what’ not ‘how’)
  – Abstract types (if any)

• Physical model focuses on physical parts and flows and normally shows ports and physical (implementation) types
  – Normally follows logical modeling
  – May be many physical models for one logical model
  – Real-world types

• May affect package structure
  – Logical package contains logical types
  – Physical package contains physical types

• Can link logical model items to physical model items via Allocation
LOGICAL DATA

`bdd [Package] Data [Logical]`

- `valueType (dataType)` Current Location
- `valueType (dataType)` Customer Data
- `valueType (dataType)` Route
- `valueType (dataType)` Navigation Instructions
- `valueType (dataType)` Credit Request
- `valueType (dataType)` Driver Input
- `valueType (dataType)` Required Location
- `valueType (dataType)` Credit Approval
- `valueType (dataType)` Traffic Data
- `valueType (dataType)` Parcel Data
- `valueType (dataType)` Power
- `valueType (dataType)` Access Request
- `valueType (dataType)` Customer Input
- `valueType (dataType)` Manifest
- `valueType (dataType)` Access Granted
EXAMPLE IBD - LOGICAL MODEL

[Image of a logical model diagram with various components and connections labeled with data flow arrows and variables such as Location Services (LS), Navigation (NV), UI Display (UD), Parcel Identification (PI), Power (PW), Credit Services (CS), UI Input (UI), User Authentication (UA), Wireless Data Transfer (WDT), and more.]
PHYSICAL DATA
INTERFACES

bdd [Package] Types [BDD]

- «interfaceBlock» IFR
- «interfaceBlock» Magnetic Reader
- «interfaceBlock» Power
- «interfaceBlock» GPS
- «interfaceBlock» Data
- «interfaceBlock» Wireless
- «interfaceBlock» Camera
EXAMPLE IBD – PHYSICAL MODEL
EXAMPLE IBD – PHYSICAL MODEL
MODEL PACKAGE STRUCTURE

• Shows Dependencies within model to interfaces
REUSING AND SHARING MODEL LIBRARIES

System A
- Logical Definitions
- Physical Definitions
- Components
  - Logical
  - Physical
  - HW
  - SW
  - Design
  - Application
  - Simulation
  - Equipment
  - UI
  - Interfaces

System B
- Logical Definitions
- Physical Definitions
- Components
  - Logical
  - Physical
  - HW
  - SW
  - Design
  - Application
  - Simulation
  - Equipment
  - UI
  - Interfaces

Library 2
- Logical Definitions
- Physical Definitions
- Interfaces
- Library 2

© 2017 PTC Permission granted to NDIA to publish and use.
ASSET-BASED DESIGN ENABLES COLLABORATION AND VIRTUAL TEAMS
ASSET-BASED MODULAR DESIGN

• Design the same way you Build
  – Construct Systems of Sub-Systems (SoS)
  – Use Services to build your Application (SOA)
  – Plug Components together (CBD)

• Modular Design
  – Top-Down, Architected
    – Specification (& Requirements) Driven
    – Parallel Working
    – Separation of Concerns
  – Bottom-Up, Asset Mining
    – Un-modeled Assets
    – Other Modeling Tools
    – Legacy Integration
    – Published Interfaces (e.g. IDL, SysML)
  – Uses the Reusable Asset Specification (RAS) and OSLC
ASSET-BASED MODULAR DESIGN

- Publish from Sub-system model into PTC Integrity Asset Library
  - Publishes the asset as a black box
  - Enables reuse as opposed to clone and own
  - Auto-creates Trace Links
ASSET-BASED MODULAR DESIGN

- Use Sub-system from PTC Integrity Asset Library in Super-system Model
- Reuse interfaces, requirements, operations, parameters, constraints, etc.
ASSET-BASED MODULAR DESIGN

• Super-system Model = Configuration of Versioned Sub-systems
THROUGH THE DEVELOPMENT LIFECYCLE
External Traces & Model Surrogates with Visual Model Trace Links

OSLC and/or URL
Drag-&-Drop for OSLC
Copy-&-Paste for URL
TRACING FROM REQUIREMENTS TO SYSML TO CAD

Right-click on items in browsers or on diagrams to open HTML Links and Surrogates

HTML Link to product data in Windchill

HTML Link to requirement in Integrity Lifecycle Manager
You define the Integrity Modeler types that are available in the ThingWorx Trace Management app.

You define the valid link types for your organization.
Trace links to all Integrity Modeler items are displayed in Windchill.

Integrity Modeler type and trace link type displayed.

Integrity Modeler icons shown.
PHYSICAL INTERFACES

Interfaces are controlled boundaries between modules, components or parts

Types include:
- Attachment, Spatial (envelope)
- Transfer (e.g. power)
- Communication
- User Interface

Transfer of Power

User Interface

Direct/Attachment

Access/Spatial

Communication
REALIZING INTERFACES

► Develop and Propagate Interfaces

Start Procedure 3

Review Interface Specification Document

Realize Interfaces with Creo Component Interfaces

End Procedure 3

In Assembly mode, add the housing then assemble select the placing component

Select Interface to Geom

Select both axes

Confirm that component has been placed correctly and repeat as necessary
COLLABORATIVE AR/VR DESIGN

A Few Simple Steps from CAD to AR/VR

Collaborate Globally

Effortlessly Collect all Relevant Information

Closed-Loop Change Management
DIGITAL TWIN

QA Engineer

Understand Your Product in the Field

Registry of Information

Identify Solutions

A digital record of each product’s designed, manufactured, serviced and real-world state

- Improve profitability by analyzing the configurations of fleets of assets for future sales, recalls or update opportunities
- Improve decision making by analyzing individual assets again their real-world usage
- Ensure security, legal and regulatory compliance with hardware and software configuration traceability
CONCLUSION

• Interface requirements start at the very beginning of development

• They are many ways to define an interface. The best one depends on particular circumstances and will change over time

• Interfaces can be traced from requirements through to architecture through to design and physical implementation

• Define common interfaces first in a collaborative environment.
  – This means they will be available when people need them.
  – They will also only be defined once

• Interfaces are where things usually go wrong so it is best to get them right.
QUESTIONS AND ANSWERS

Thanks for your attention!

Speaker