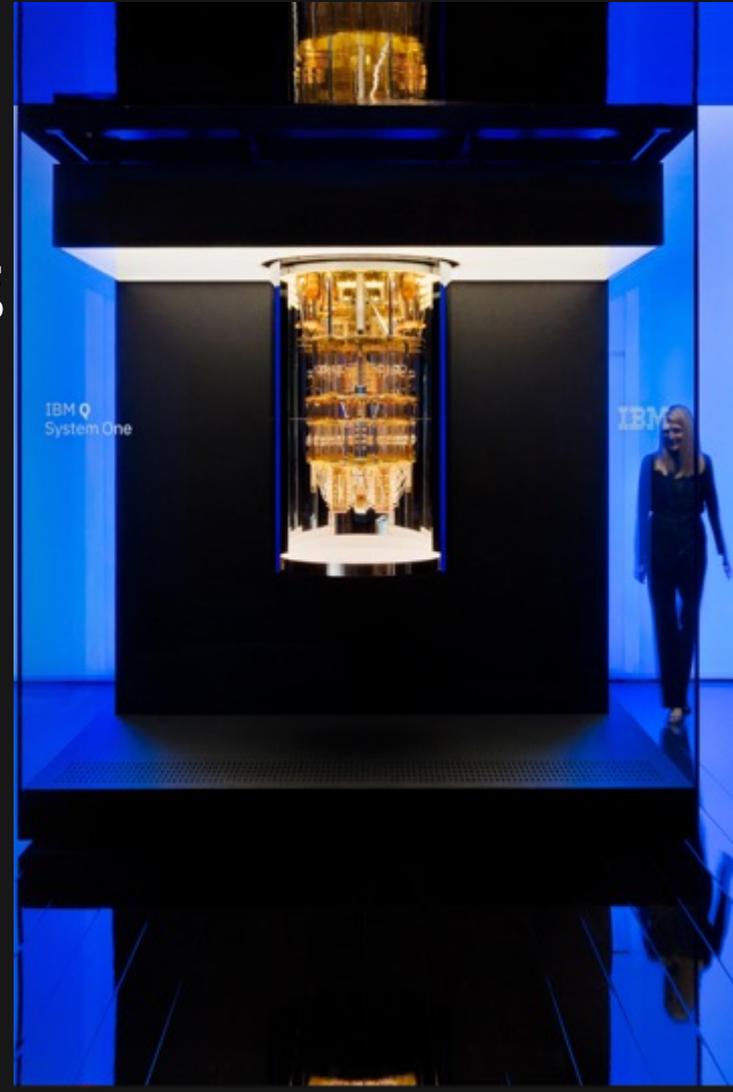


Digital Engineering

Requirements to Systems Engineering with HarmonyMBE



Peter Schedl
Program Manager
IBM Engineering Lifecycle Management
peter.schedl@de.ibm.com



Questions to be answered

- What is Requirements Engineering ?
- Does Modeling help ?
- How much Modeling do I need?
- What is MBSE and how does it relate to Requirements Engineering ?
- What is HarmonyMBE ?
- How to validate Requirements - or - what is this ?



Requirements Definition and Management

Requirements Engineering involves:

- Requirements elicitation – gathering requirements from stakeholders/customers
- Requirements analysis and negotiation – check clarity, completeness, resolve issues
- Requirements specification – document with text, maybe with use cases and scenarios
- Functional analysis
- Requirements validation
- Requirements management – continuous activity, traceability, change handling

The screenshot displays a requirements management interface. On the left, a 'Contents' pane shows a hierarchical tree of requirements:

- 1 Adaptive Cruise Control initial requirements
 - 1.1 Basic functionality
 - 1.1.1 Safety Goals
 - 1.1.1.1 Safe Cruise Control Deactivation
 - 1.2 Operating Modes
 - 6549 The two main operating modes of the ACC
 - 6550 Speed control mode shall employ basic
 - 6552 Time gap mode shall be based upon a
 - 6553 The time gap shall be maintained by a

The main area shows a detailed view of requirement 6551: UAV. The requirement text is: "We want to achieve a multi use UAV. It shall be the fastest and long ranged s. It should have a civil flight allowance. We want it to be easily serviceable." A 'Show More' button is visible below the text. A tooltip or detail view is open, showing the full text: "An unmanned aerial vehicle (UAV), commonly known as a drone, is an aircraft without any human pilot, crew, or passengers on board."

Requirements Challenges

Misunderstood requirements by stakeholders and engineers

Poorly expressed requirements → [RQA: AI based quality checker](#)

Misunderstanding or omission by development

Missed test coverage

Requirement change impact misunderstandings

Little reuse

The screenshot displays a user interface for an AI-based quality checker. At the top, it shows 'Quality scores (0 - 100)' and '2 artifacts selected'. Below this, there is a list of requirements with their respective quality scores: '72483The GPS shall, where there is sufficient space, di...' with a score of 70, and '72481The GPS system shall provide a clear perspective ...' with a score of 80. A blue arrow points from the second requirement to a detailed view. This view includes the text 'Unclear term', a suggestion to 'Look for: clear', and a 'Teach Watson' button. The detailed view also contains a paragraph explaining that the requirement includes a term or phrase that makes it unclear and provides guidance on how to improve it by using specific and precise terms.

Quality scores (0 - 100)
2 artifacts selected [Recheck these artifacts](#)

- 72483The GPS shall, where there is sufficient space, di... 70
- 72481The GPS system shall provide a clear perspective ... 80

72481The GPS system shall provide a clear perspective of the road. 80

Unclear term
Look for: clear
[Hide details](#) [Teach Watson](#)

This requirement includes a term or phrase that makes the requirement unclear. Effective requirements use specific and precise terms that have only one interpretation. Replace vague and general terms with specific, descriptive terms or rewrite the requirement with different wording to ensure that all phrases are clear.

Requirements Challenges

Misunderstood requirements by stakeholders and engineers

Poorly expressed requirements  ChatGPT example

Misunderstanding or omission by development

Missed test coverage

Requirement change impact misunderstandings

Little reuse

“Ok. That’s what we need to do:

Thread A will pass event X to thread B and that will change B’s state to Running from what it was before which was Idle. When B changes to Running it will send back an event Y to A and then wait for 2 second and then go back to Idle. Thread A will have started in Idle also and will go to Run after B sends back event Z which happens after the 2 seconds before going to Idle. All this should happen in less then 5 seconds.”



Functional Requirements:

- 1.Thread A shall pass event X to Thread B.
- 2.Upon receiving event X, Thread B shall change its state from Idle to Running.
- 3.After changing to Running state, Thread B shall send event Y to Thread A.
- 4.Thread B shall wait for 2 seconds after sending event Y.
- 5.After waiting for 2 seconds, Thread B shall change its state from Running to Idle.
- 6.Thread A shall start in Idle state.
- 7.After receiving event Y from Thread B, Thread A shall change its state from Idle to Running.
- 8.Thread A shall wait for event Z before changing its state to Idle.
- 9.Event Z shall be sent by Thread B after waiting for 2 seconds in Running state.
- 10.The entire process of passing event X from Thread A to Thread B, changing states, waiting

Non-Functional Requirements:

- 1.The system shall ensure that the communication between Thread A and Thread B is reliable
- 2.The system shall ensure that the time delay between sending events, changing states, and
- 3.The system shall be designed to handle multiple instances of the process simultaneously w

Requirements Challenges

Misunderstood requirements by stakeholders and engineers

Poorly expressed requirements

Misunderstanding or omission by development

Missed test coverage

Requirement change impact misunderstandings

Little reuse



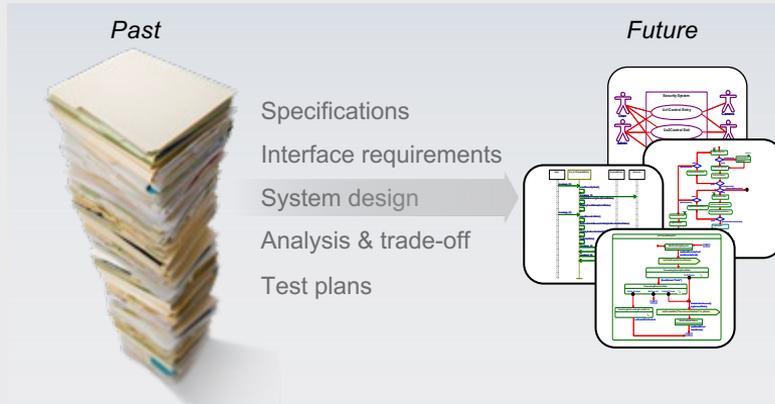
“Ok. That’s what we need to do:

Thread A will pass event X to thread B and that will change B’s state to Running from what it was before which was Idle. When B changes to Running it will send back an event Y to A and then wait for 2 second and then go back to Idle. Thread A will have started in Idle also and will go to Run after B sends back event Z which happens after the 2 seconds before going to Idle. All this should happen in less then 5 seconds.”

Modeling in Requirements Engineering

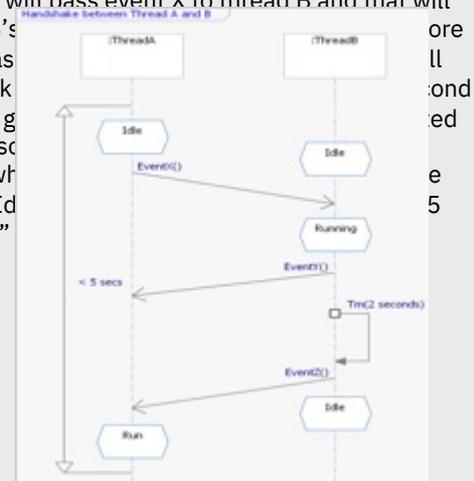
Requirements Engineering involves:

- Requirements elicitation
- Requirements analysis and negotiation
- **Requirements specification**
- **Functional analysis**
- **Requirements validation**
- Requirements management



“Ok. That’s what we need to do:

Thread A will pass event X to thread B and that will change B’s state from Idle to Running and then go back to Idle also. Thread A will then pass event Z which will go to Idle state in 5 seconds.”



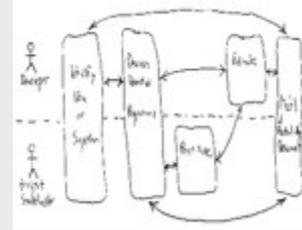
Why Modeling?

Manage complexity

- Complicated applications need a visual plan

Simplify and abstract ! essential aspects of a system

- Increase understanding of requirements



Enhance communication

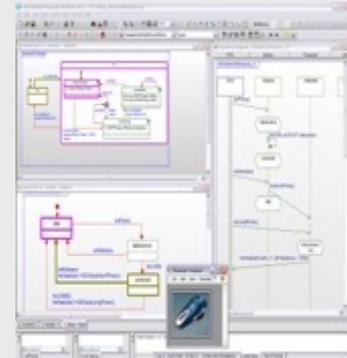
- Common language promotes common understanding across disciplines

Reduce risk

- Model execution increases knowledge and reduces uncertainty and risk

Provide traceability

- Models document what you have done

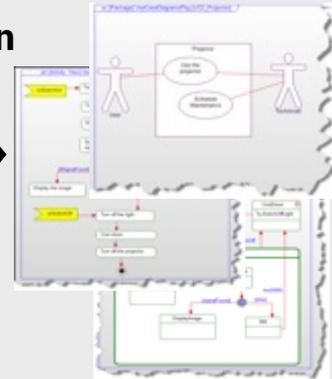
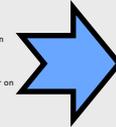


Modeling in Requirements Engineering

Requirements Engineering involves:

- Requirements elicitation
- Requirements analysis and negotiation
- **Requirements specification – use cases and scenarios for describing user interactions**
- **Functional analysis – functional flows, interface definition, documented rationale**
- **Requirements validation – testing of requirements through model execution**
- Requirements management

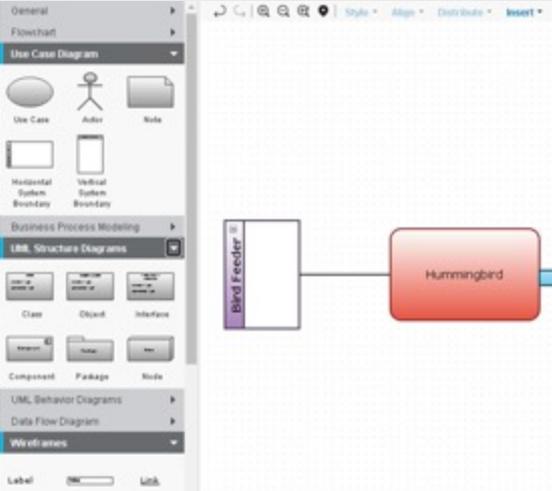
2 Functional Requirements
2.1 Power car
2.1.1 Move car
2.1.1.1 Move forwards
The car shall be able to move forwards at all speeds from 0 to 200 kilometers per hour on standard flat roads with winds of 0 kilometers per hour, with 180 BHP.
2.1.1.2 Move backwards
The car shall be able to move backwards to a maximum speed of 20 Kilometers per hour on standard flat roads with winds of 0 kilometers per hour, with 180 BHP.
2.1.2 Accelerate car
The car shall be able to accelerate from 0 to 100 Kilometers per hour in 10 seconds on standard flat roads with winds of 0 kilometers per hour.
The car shall be able to accelerate from 100 to 150 kilometers per hour at a rate of 5 kilometers per second on standard flat roads with winds of 0 kilometers per hour.
The car shall be able to accelerate from 150 to 200 kilometers per hour at a rate of 3 kilometers per second on standard flat roads with winds of 0 kilometers per hour.
2.2 Control car
2.2.1 Switch on car
The car shall be able to discriminate which authorized people shall be able to switch on and operate the car.
2.2.2 Control speed
The car shall have a foot mechanism to control the speed of the car.
The speed control shall be infinitely variable from zero to maximum speed.



Improve Requirements specification by just adding “Diagrams”

Fully integrated (like textual requirement)

Offers rich notations



Not a formal model !

4136 -1.1 Vision

4112 The *Aviary* is a system of systems consisting of a pilot controller (known as the *Bird Feeder*), a UAV (known as the *Hum* computers).

4067

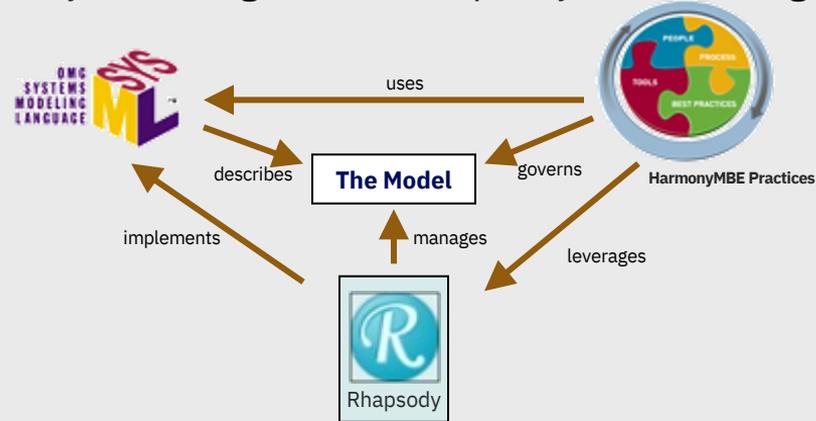
4073 The Hummingbird is a small drone use for surveillance with a short range and flight. It is intended to have:

The diagram shows a vertical rectangle labeled 'Bird Feeder' connected by a line to a rounded rectangle labeled 'Hummingbird'. A blue arrow points from the 'Hummingbird' box to a larger box labeled 'Bird Watchers'. Inside the 'Bird Watchers' box are four green smartphone-like icons arranged in a 2x2 grid.

IBM's Model Based Systems Engineering (MBSE) Solution

MBSE is a standards based Systems Engineering practice that incorporates:

- Modeling language – SysML
- Modeling method – Harmony Systems Engineering Practices
- Modeling tool – Rhapsody for Systems Engineers & Rhapsody Model Manager

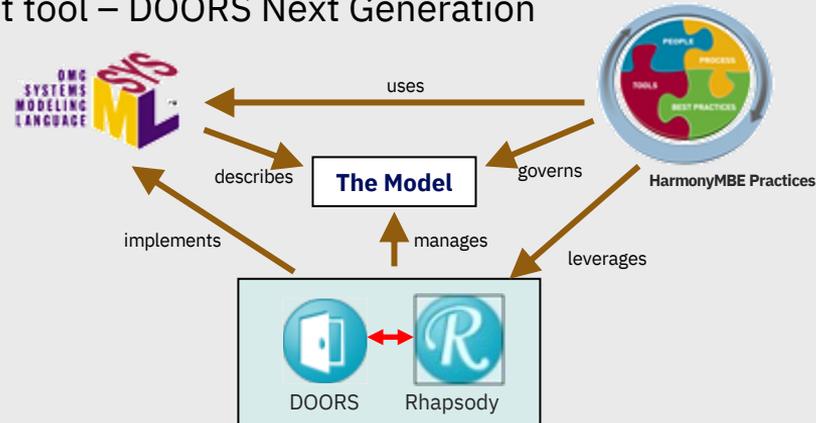


Requirements ?

IBM's Model Based Systems Engineering (MBSE) Solution

MBSE is a standards based Systems Engineering practice that incorporates:

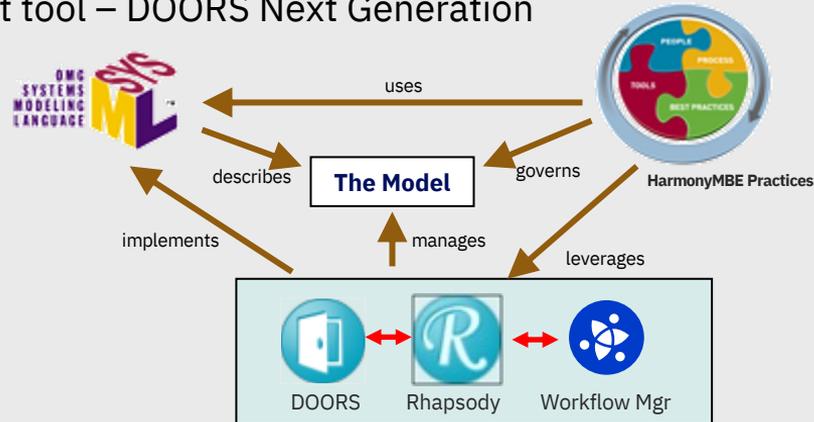
- Modeling language – SysML
- Modeling method – Harmony Systems Engineering Practices
- Modeling tool – Rhapsody for Systems Engineers & Rhapsody Model Manager
- Requirements management tool – DOORS Next Generation



IBM's Model Based Systems Engineering (MBSE) Solution

MBSE is a standards based Systems Engineering practice that incorporates:

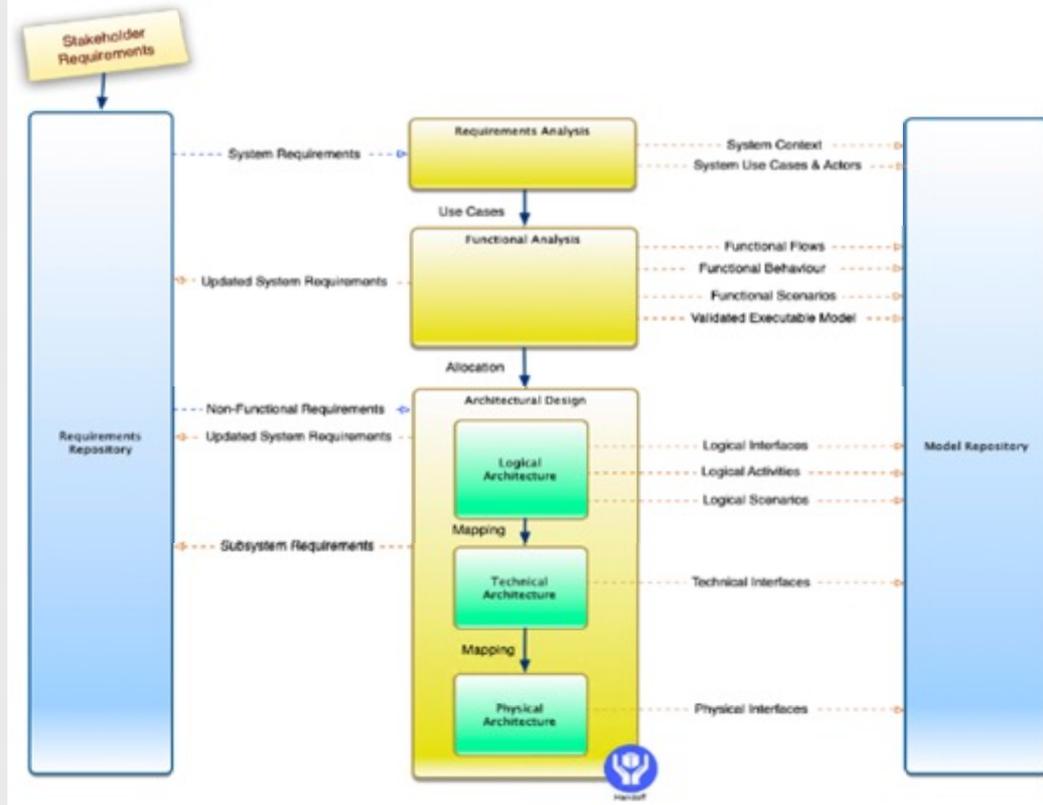
- Modeling language – SysML
- Modeling method – Harmony Systems Engineering Practices [incl. Ticket System for Guidance & Com.](#)
- Modeling tool – Rhapsody for Systems Engineers & Rhapsody Model Manager
- Requirements management tool – DOORS Next Generation



HarmonyMBE Practice provides Guidance & Automation

Model Based Systems Engineering complements traditional requirements analysis techniques

- during Requirements Analysis, we organize requirements into functional groups (use cases)
- during Functional Analysis, we identify system functions and explore the system's dynamic behavior using activity diagrams and model execution
- during Architectural Design, system operations are allocated to decomposed Logical Architecture
- Finally either direct Hand-Off to the engineering teams or first map to a Technical Architecture



HarmonyMBE Workflow – Integrated in Tool Rhapsody

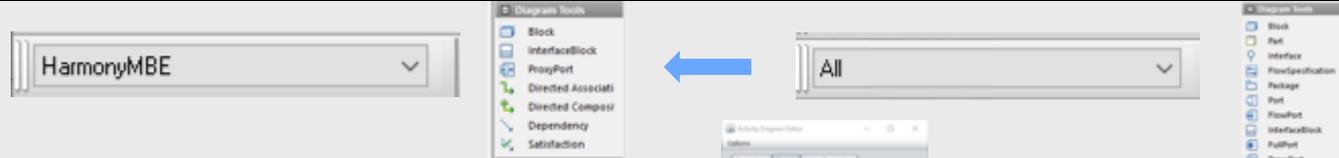
HarmonyMBE Workflow - Can be used with any Tool

The screenshot displays the IBM Engineering Systems: Design Rhapsody HarmonyMBE - Demo interface. The main window is titled "Welcome to Rhapsody" and features a navigation bar with tabs for Overview, Requirement Analysis, Functional Analysis, Logical Architecture, Technical Architecture, and Physical Architecture. The Overview tab is active, showing a "Welcome to Harmony MBE" message and a diagram of the canonical workflow phases. The diagram illustrates the flow from Requirements Analysis to Functional Analysis, Logical Architecture, Technical Architecture, and Physical Architecture, with various sub-phases and artifacts. A yellow circle highlights the "Requirements Analysis Checklist" on the right side of the interface. The checklist includes sections for Introduction, Import Requirements, Create and Trace Use Cases, and Next - Functional Analysis. The "Introduction" section states: "This Guide-Me walks you through a simplified task list for the Requirements Analysis phase. Note that this is only a simple checklist, please refer to the HarmonyMBE Desk Book for detailed guidance. This is also based on the canonical workflow; alternative workflows are available." The "Import Requirements" section mentions: "Import requirements using either OSLC or the CSV importer provided." The "Create and Trace Use Cases" section includes tasks like "Define Actors:", "Populate the Use Case Diagram:", and "Add Traceability:". The "Next - Functional Analysis" section is partially visible. The left sidebar shows a tree view of the model structure, including Packages, ActorPkg, DemoEventsPkg, FunctionalAnalysisPkg, and RequirementsAnalysisPkg. A yellow circle also highlights the "Requirements Repository" in the left sidebar.

IBM Engineering / © 2023

HarmonyMBE simplifies MBSE

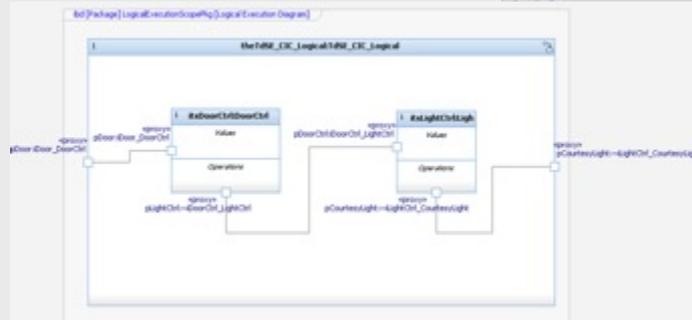
Reduced Menu's & Toolbars



Model creation Support (e.g. Activity Diagram Editor)



Auto generation of ports, interfaces, delegation ports, connectors and diagrams



New architecture layers are generated on demand

- Allocation & Re-use of blocks, ports and interfaces

Harmony MBE Benefits

Proven Method with simplified Tool UI and automation

Only create core information: Use Cases, Activities and Subsystems (BDD) incl. State Charts



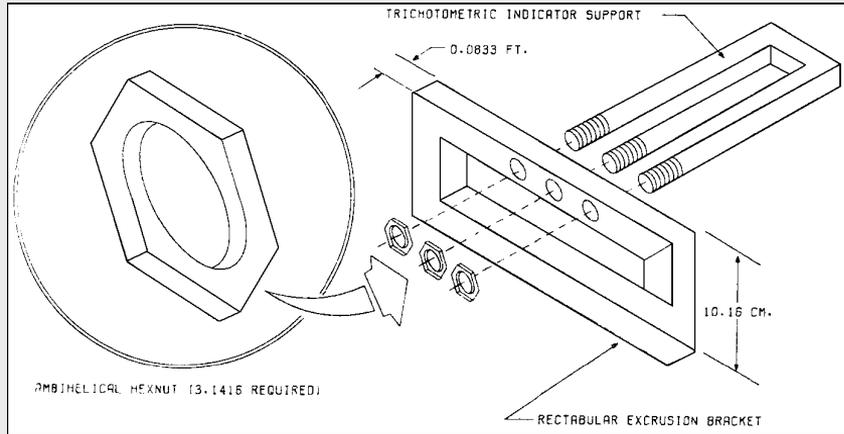
Get verified System Architecture incl. Interfaces (ICD) & Test Scenarios



Linked to Requirements & Tasks

Is my Model correct ?

„How do you validate that you have a ‚useful‘ model?



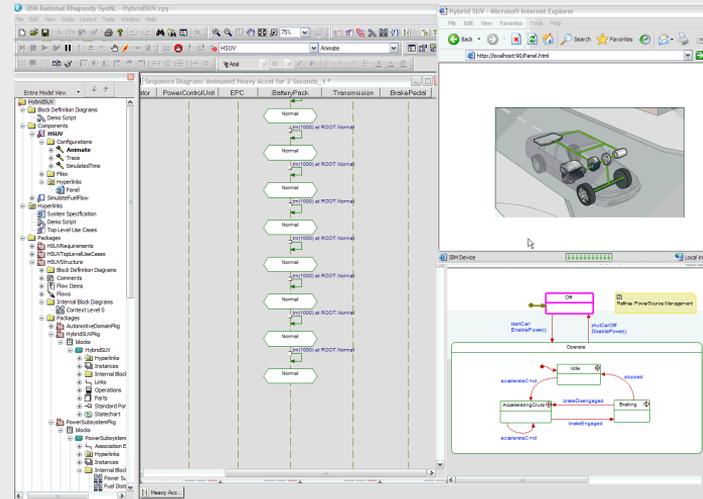
Continuously validate and verify thru Model Execution

Once the system behavior has been captured it can be verified through execution

- The Sequence Diagrams should be used as the basis for stimulating the model and to record the test run

Advantages of execution

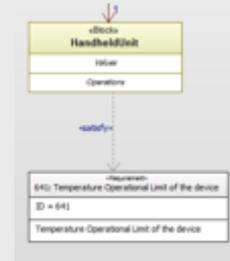
- Helps eliminate bugs and design flaws
- Helps build a robust system as unhandled conditions are exposed
- Helps test interfaces...Reducing integration issues further down
- Helps test for regressions
- Verified Test Cases can be handed down the life-cycle



Requirements Revisited – Modellers Point of View

Functional Requirements have been traced to Model Elements like Use Cases, Blocks, Operations, Interface Data

Non-functional requirements are traced to Subsystem Blocks



Visualize the Traceability in a Matrix View

From: block, PrimitiveOperation		Scope: ParkingCtrlPkg	
	<input type="checkbox"/> ParkingCtrl	<input checked="" type="checkbox"/> confirmGuest	
<input checked="" type="checkbox"/> SR001 - Entry		<input checked="" type="checkbox"/> SR001 _ Entry	
<input checked="" type="checkbox"/> SR007 - Allocation of Parking Spaces		<input checked="" type="checkbox"/> SR007 _ Allocation of Parking Spaces	
<input checked="" type="checkbox"/> SR009 - Price Information	<input checked="" type="checkbox"/> SR009 _ Price Information		
<input checked="" type="checkbox"/> SR014 - Pay via Checkout		<input checked="" type="checkbox"/> SR014 _ Pay via Checkout	

When a Requirement changes – the impact of that change may be analyzed

Requirements traced to Model Artefacts – RE Point of View

Requirements tight integration with Models



Join also my other talk about Importance of Traceability

ID	Contents	Traced By Architecture Element
6546	The system shall detect the opening of a side door.	UseCase: Control Courtesy Light
6542	The system shall detect the opening of the trunk.	UseCase: Control Courtesy Light
6504	The system shall detect the closing of a side door.	UseCase: Control Courtesy Light
6516		UseCase: Control Courtesy Light
6530		UseCase: Control Courtesy Light
6543	UseCase: Control Courtesy Light Project Name: Automotive Development Name: Control Courtesy Light Type: UseCase Last Modified: One hour ago	UseCase: Control Courtesy Light
6541		UseCase: Control Courtesy Light
6522		UseCase: Control Courtesy Light
6508		UseCase: Control Courtesy Light

When a Requirement changes – the impact of that change may be analyzed

Summary

Model Based Systems Engineering (MBSE) complements traditional requirements definition and management techniques

- Work at appropriate level of abstraction graphically
- Can be scaled from light “drawing” to full MBSE incl. model simulation
- Continuously validate and verify
- Provide detailed traceability

IBM provides with HarmonyMBE a SysML-based Systems Engineering practice library with application guidance

Functional Analysis ...

- is a Requirements Engineering activity
- emphasizes the transformation of functional system requirements into a coherent description of system functions
- improves upon the quality of system requirements

Thank You



Peter Schedl
peter.schedl@de.ibm.com

Further information:

[IBM Engineering Lifecycle Management Automotive Compliance](#)

[IBM Engineering Management Overview](#)

© Copyright IBM Corporation 2023. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and ibm.com are trademarks of IBM Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at [Copyright and trademark information](#).

