

# Agile Model-Based Systems Engineering (aMBSE)

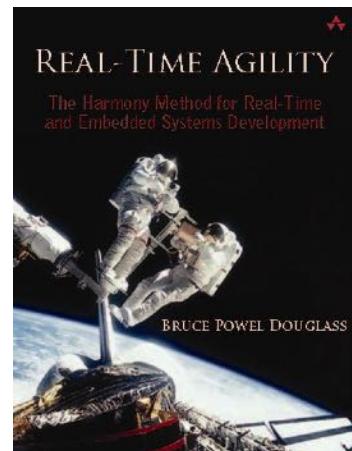
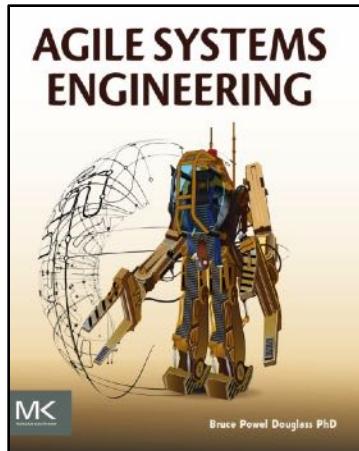
Bruce Powel Douglass, Ph.D.

Chief Evangelist, Global Technology Ambassador  
IBM IoT

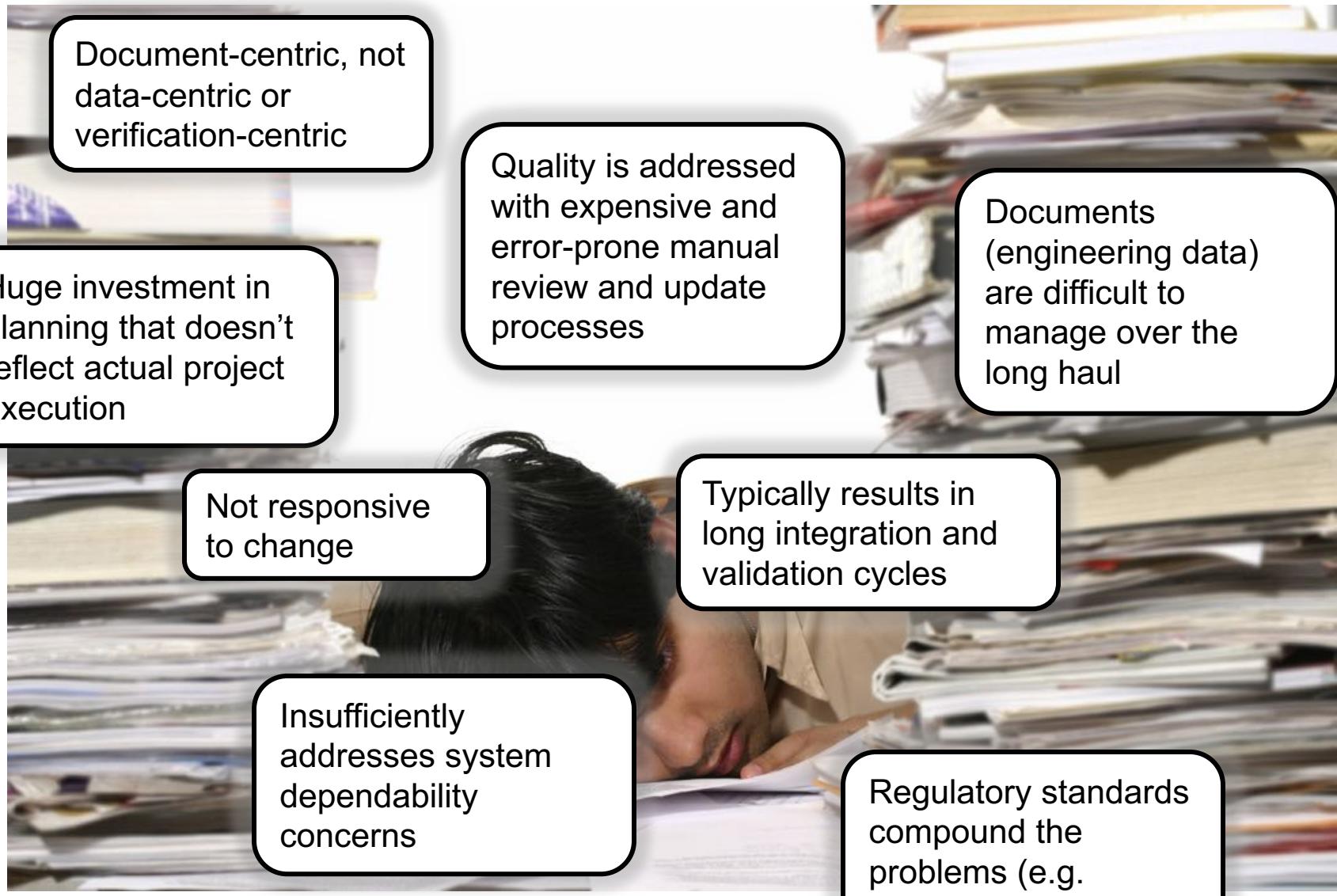
[Bruce.Douglass@us.ibm.com](mailto:Bruce.Douglass@us.ibm.com)

Twitter: @BruceDouglass

IBM: [www-01.ibm.com/software/rational/leadership/thought/BruceDouglass.html](http://www-01.ibm.com/software/rational/leadership/thought/BruceDouglass.html)



# State of the Practice for Systems Development



# Key Concepts for Agility

## Improve quality through *continuous feedback*

### Verification

Formal analysis

Review

Testing via execution or simulation

**Validation:** Customer feedback (*meet the need*)

Correctness

Appropriateness

Usability

**Dependability:** Analysis of safety, reliability, & security

**Primarily build executable things**

**Verify them continuously**

**This means MODELS**

## Efficiency through

Concentrate on high-value tasks

Avoid rework

Paying attention to how you're doing against goals

Project retrospective

Risk management

**Active and continuous risk mitigation**  
**Monitor project success**

## Planning

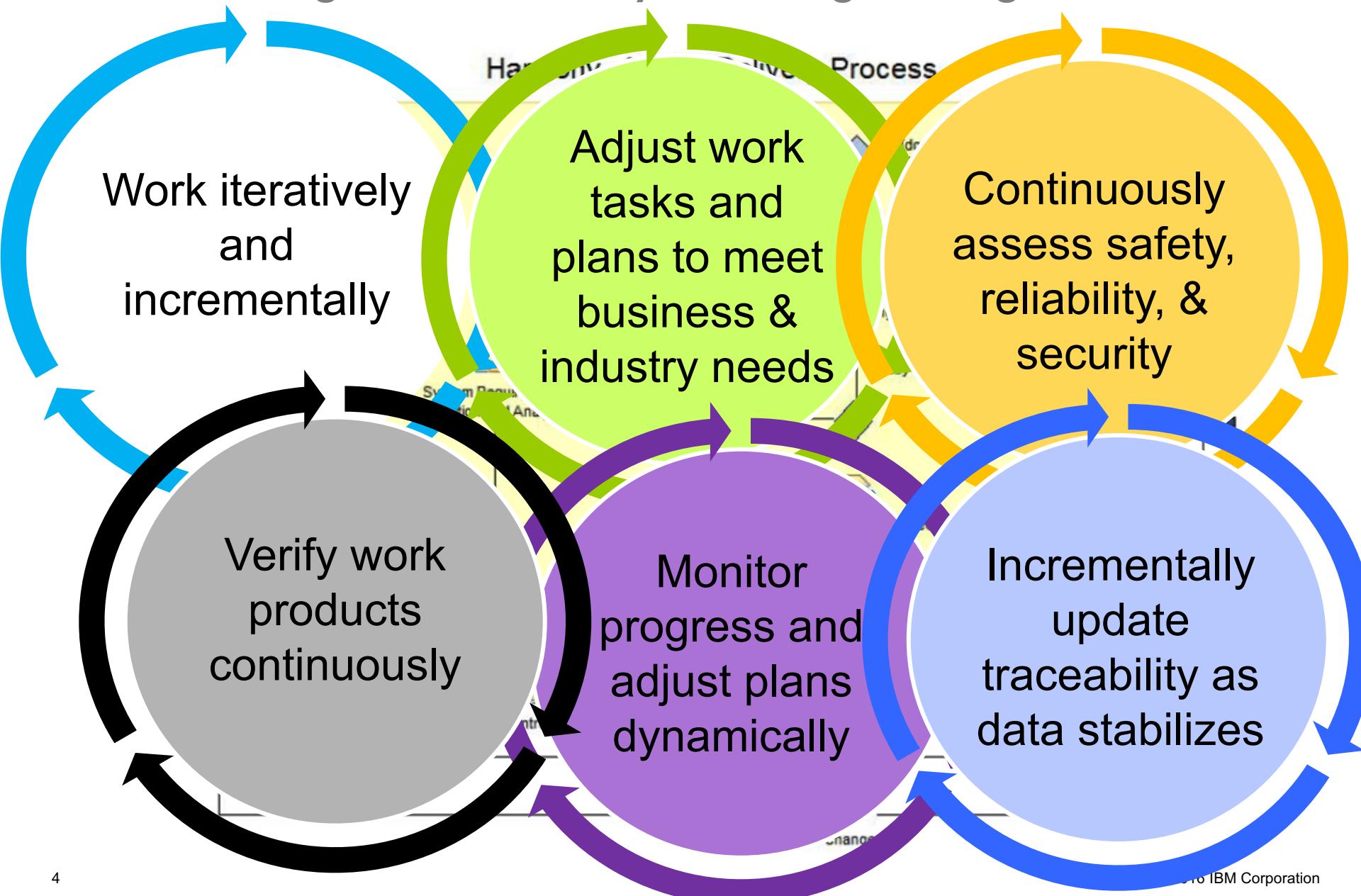
Don't plan beyond the fidelity of the information you have

Plan enough but not more than that

Adjust plans based on "truth on the ground" (metrics)

**Dynamic planning**  
**Responsive to Change**

# What does “agile” mean for Systems Engineering?



# Computable models are essential for Agile MBSE

# precision

# network

# accuracy

# verification

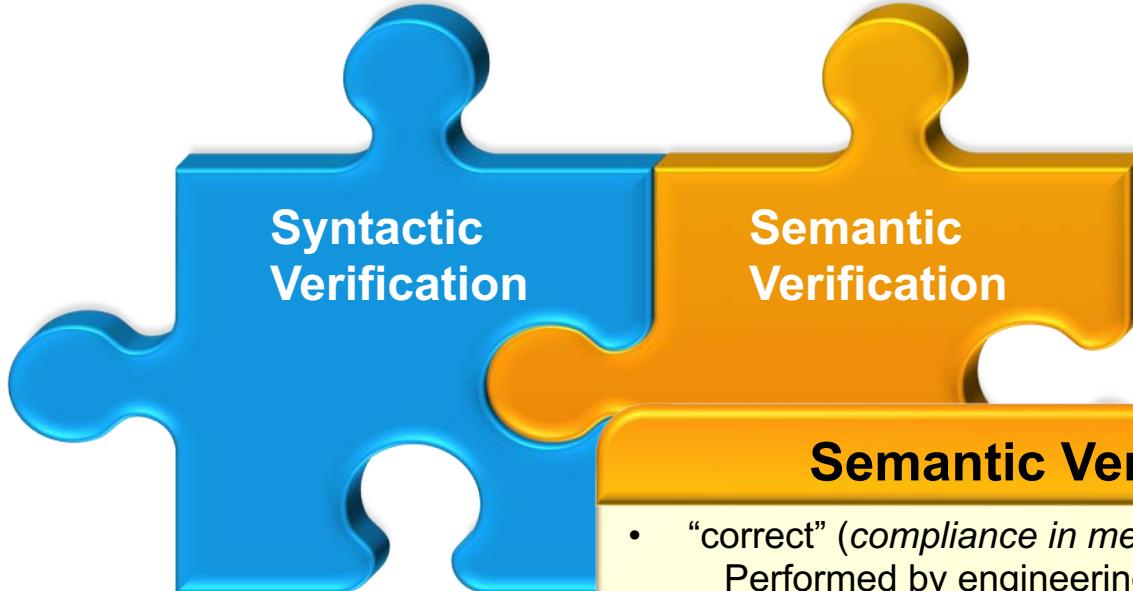
# coverage

# validation

# neural dynamics

consistency

# What do we mean by “verification”?



## Syntactic Verification

- “well-formed” (*compliance in form*)  
Performed by quality assurance personnel
- **Audits** – work tasks are performed as per plan and guidelines
- **Syntactic review** – work products conform to standard for organization, structure and format

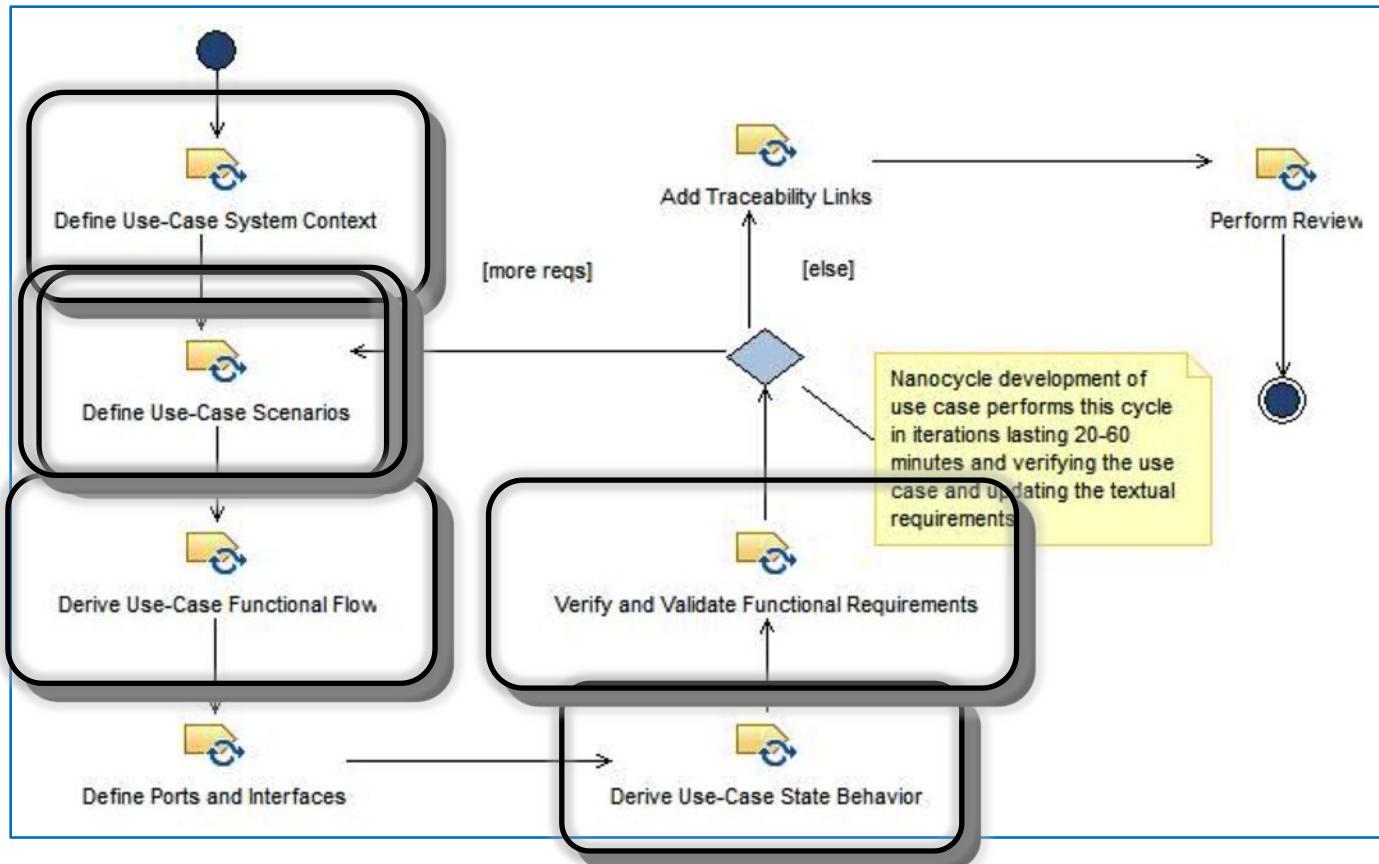
## Semantic Verification

- “correct” (*compliance in meaning*)  
Performed by engineering personnel

Three basic techniques

- **Testing** – requires executability of work products, impossible to fully verify
- **Formal methods** – strongest but hard to do and subject to invariant violation
- **Semantic review** (subject matter expert & peer) – most common, weakest means

# Scenario Driven Use Case Construction / Verification



## Making it Agile

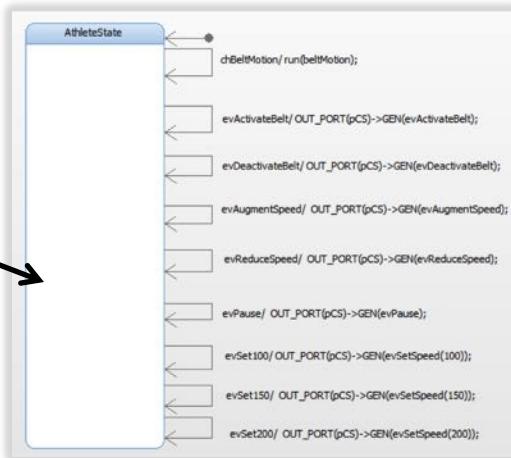
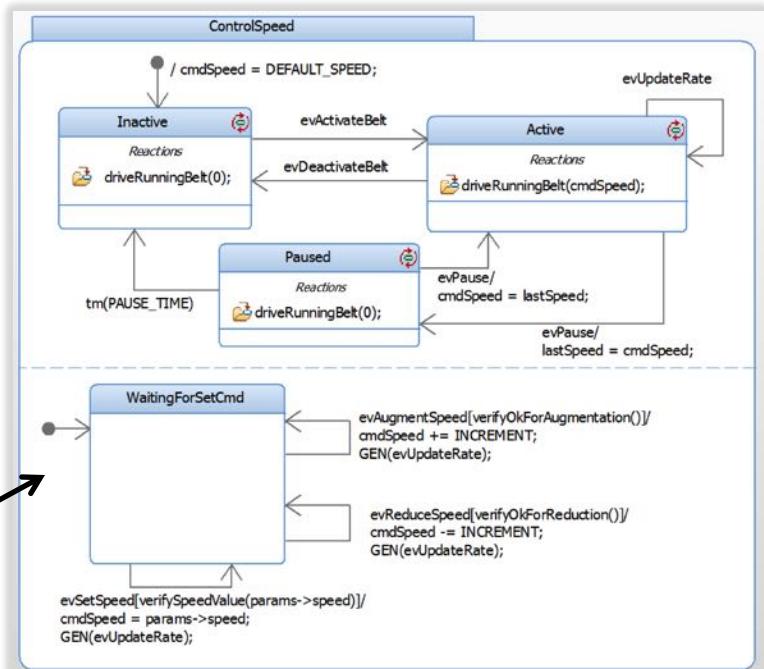
Loop

Loop  
< 1 hr

Repeat until all requirements added  
Repeat for all use cases

Conceptualize requirement aspect  
Incrementally augment model  
Verify

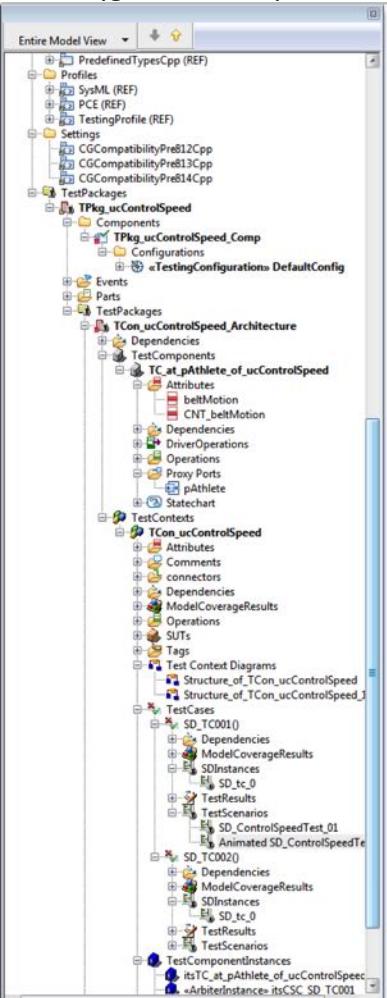
# High Fidelity Models are verifiable



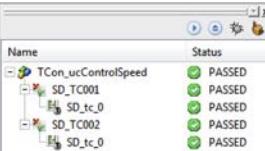
... and may be built and verified incrementally, enabling agility

# Verification may be manually done or via Test Conductor

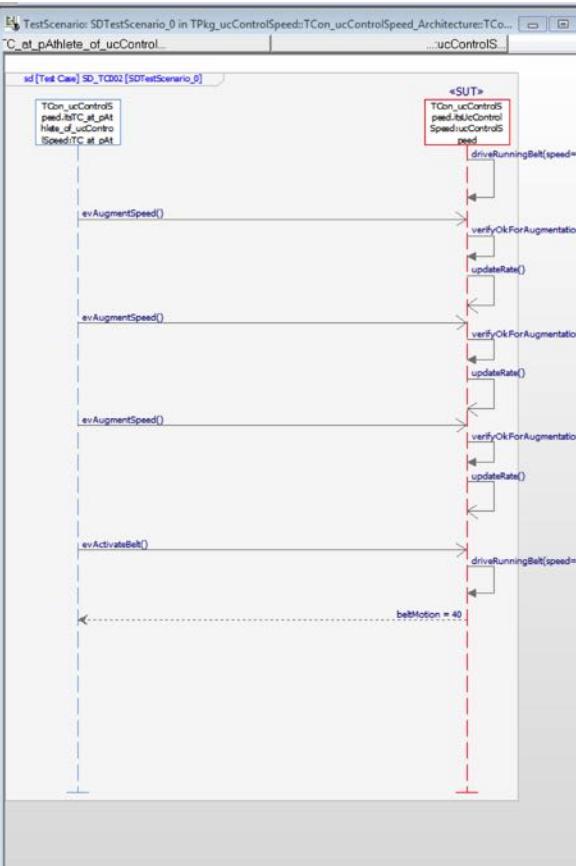
## Test Architecture (generated)



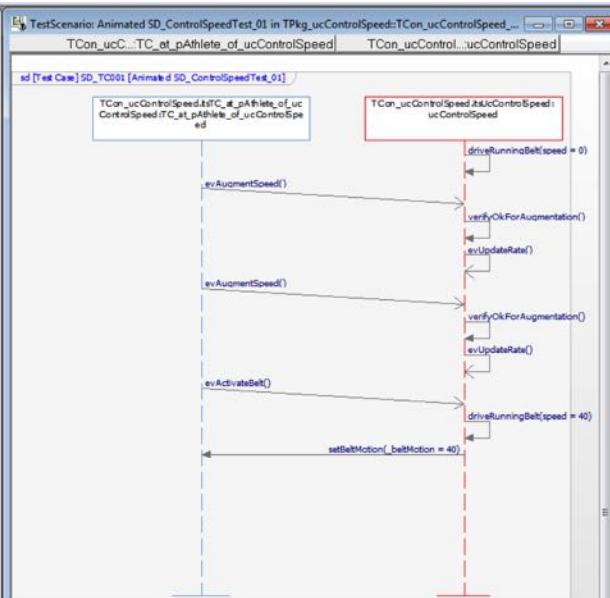
## Test outcome



## Test case definition



## Test case test case result



Test Conductor implements the OMG UML Testing Profile standard

# Determine the degree of verification with model coverage analysis

**TestCase Coverage Result**

TestCase: SD\_TC001

Environment Information	
Test executed on machine:	P8050Z6-27298
Test executed by user:	Bruce Douglass
Used operating system version:	Windows 7
Used Rhapsody version:	8.1.4, build 9641846
Used TestConductor version:	2.6.4, build 4386
Generation Date:	Wednesday, April 13, 2016 15:18:58

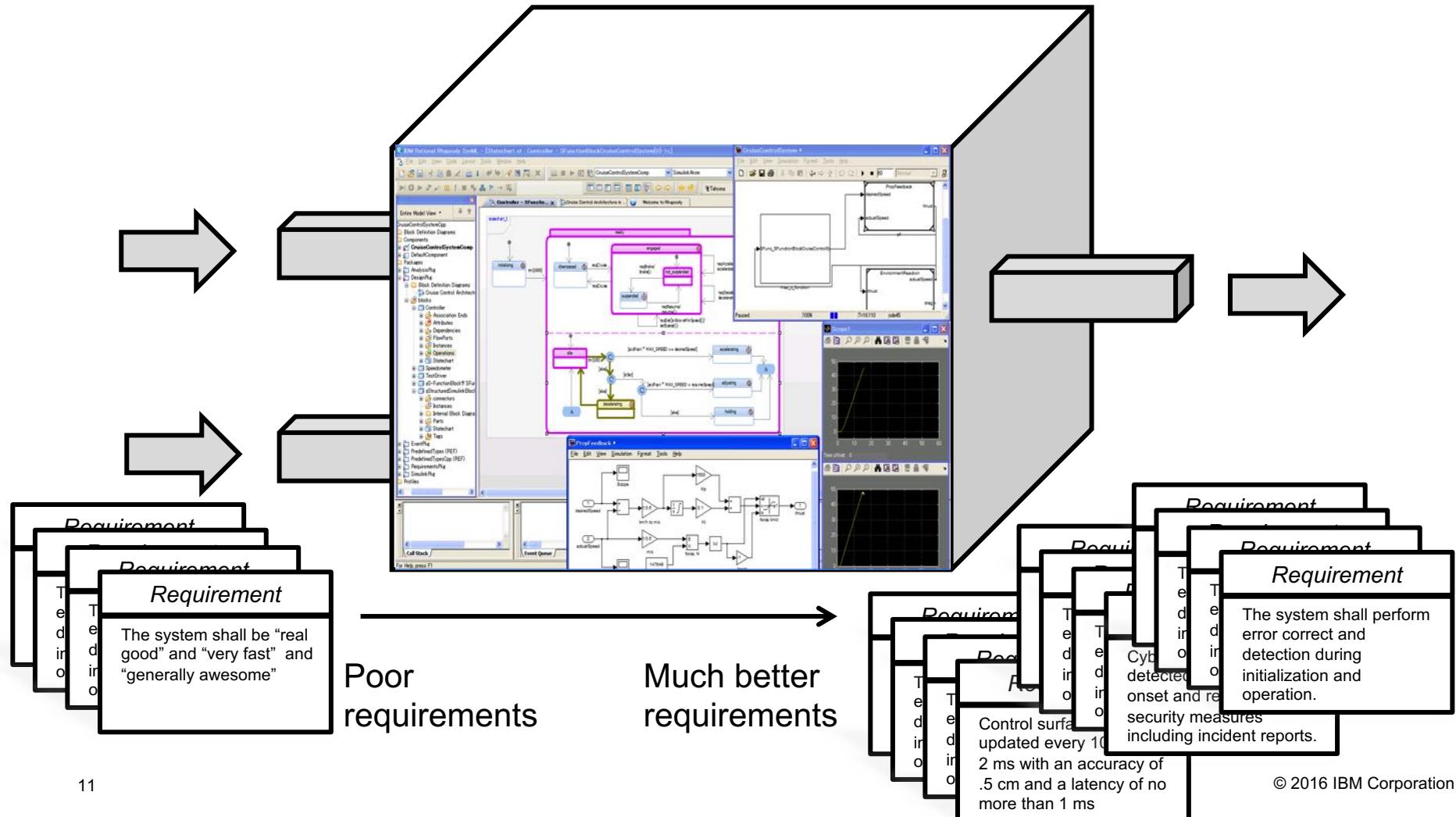
Tested Project	
Project:	SpeedDemonTreadmill
Active Code Generation Component:	TPkg_ucControlSpeed_Comp
Active Code Generation Configuration:	DefaultConfig
TestPackage:	TPkg_ucControlSpeed::TCon_ucControlSpeed_Architecture
TestContext:	TCon_ucControlSpeed
TestCase:	SD_TC001
Covered Classes:	ucControlSpeed (highlight   goto)

Detailed Coverage Summary of ucControlSpeed (15/29)	
<ul style="list-style-type: none"> <li>Operations           <ul style="list-style-type: none"> <li>covered <a href="#">driveRunningBelt</a></li> <li>covered <a href="#">verifyOkForAugmentation</a></li> <li>not covered <a href="#">verifyOkForReduction</a></li> <li>not covered <a href="#">verifySpeedValue</a></li> </ul> </li> <li>EventReceptions           <ul style="list-style-type: none"> <li>covered <a href="#">evAugmentSpeed</a></li> <li>not covered <a href="#">evReduceSpeed</a></li> <li>not covered <a href="#">evSetSpeed</a></li> <li>covered <a href="#">evActivateBelt</a></li> <li>not covered <a href="#">evDeactivateBelt</a></li> <li>covered <a href="#">evUpdateRate</a></li> <li>not covered <a href="#">evPause</a></li> </ul> </li> <li>StateChart: statechart_2 of ClassucControlSpeed           <ul style="list-style-type: none"> <li>covered <a href="#">ROOT_ControlSpeed</a></li> <li>covered <a href="#">ROOT_ControlSpeed.state_6</a></li> <li>covered <a href="#">ROOT_ControlSpeed.state_6.Active</a></li> <li>covered <a href="#">ROOT_ControlSpeed.state_6.Inactive</a></li> <li>not covered <a href="#">ROOT_ControlSpeed.state_6.Paused</a></li> <li>not covered <a href="#">10</a></li> <li>covered <a href="#">3</a></li> <li>not covered <a href="#">4</a></li> <li>covered <a href="#">5</a></li> <li>not covered <a href="#">2</a></li> <li>not covered <a href="#">8</a></li> <li>not covered <a href="#">9</a></li> <li>covered <a href="#">ROOT_ControlSpeed.state_7</a></li> <li>covered <a href="#">ROOT_ControlSpeed.state_7.WaitingForSetCmd</a></li> <li>covered <a href="#">0</a></li> <li>not covered <a href="#">1</a></li> <li>not covered <a href="#">2</a></li> <li>covered <a href="#">6</a></li> </ul> </li> </ul>	
<input type="checkbox"/> hide covered <a href="#">Go to: class top</a>	
<a href="#">Operation</a>	
<a href="#">EventReception</a>	
<a href="#">Composite State</a>	
<a href="#">Composite State</a>	
<a href="#">State</a>	
<a href="#">State</a>	
<a href="#">State</a>	
<a href="#">Transition</a>	
<a href="#">Transition</a>	
<a href="#">Transition</a>	
<a href="#">Default Transition</a>	
<a href="#">Transition</a>	
<a href="#">Transition</a>	
<a href="#">Transition</a>	
<a href="#">Composite State</a>	
<a href="#">State</a>	
<a href="#">Transition</a>	
<a href="#">Transition</a>	
<a href="#">Default Transition</a>	

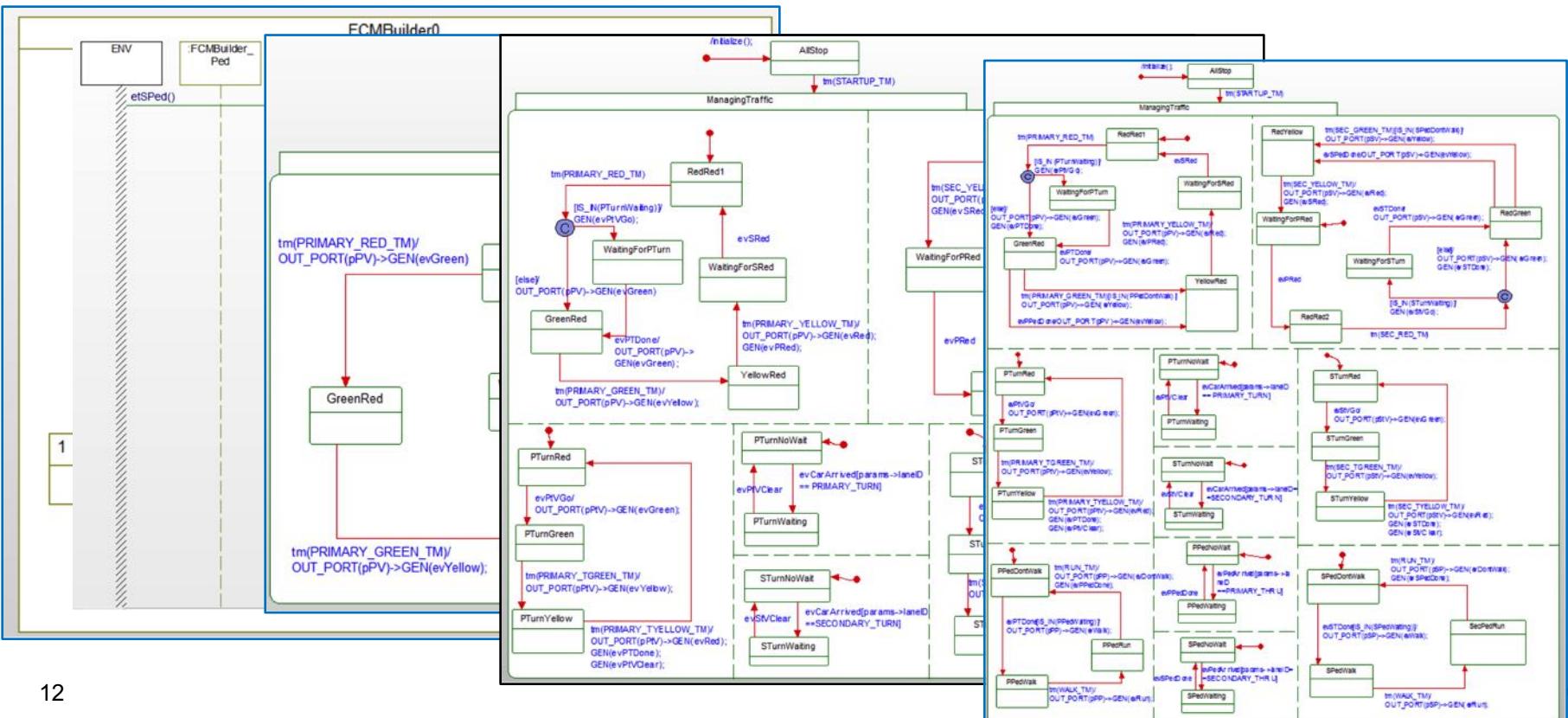
# Functional Analysis via Executable Requirements?

- A functional requirement is a *specification of an input-output control or data transformation or flow*
- A quality of service requirements is a specification of how well a control or data transformation is achieved



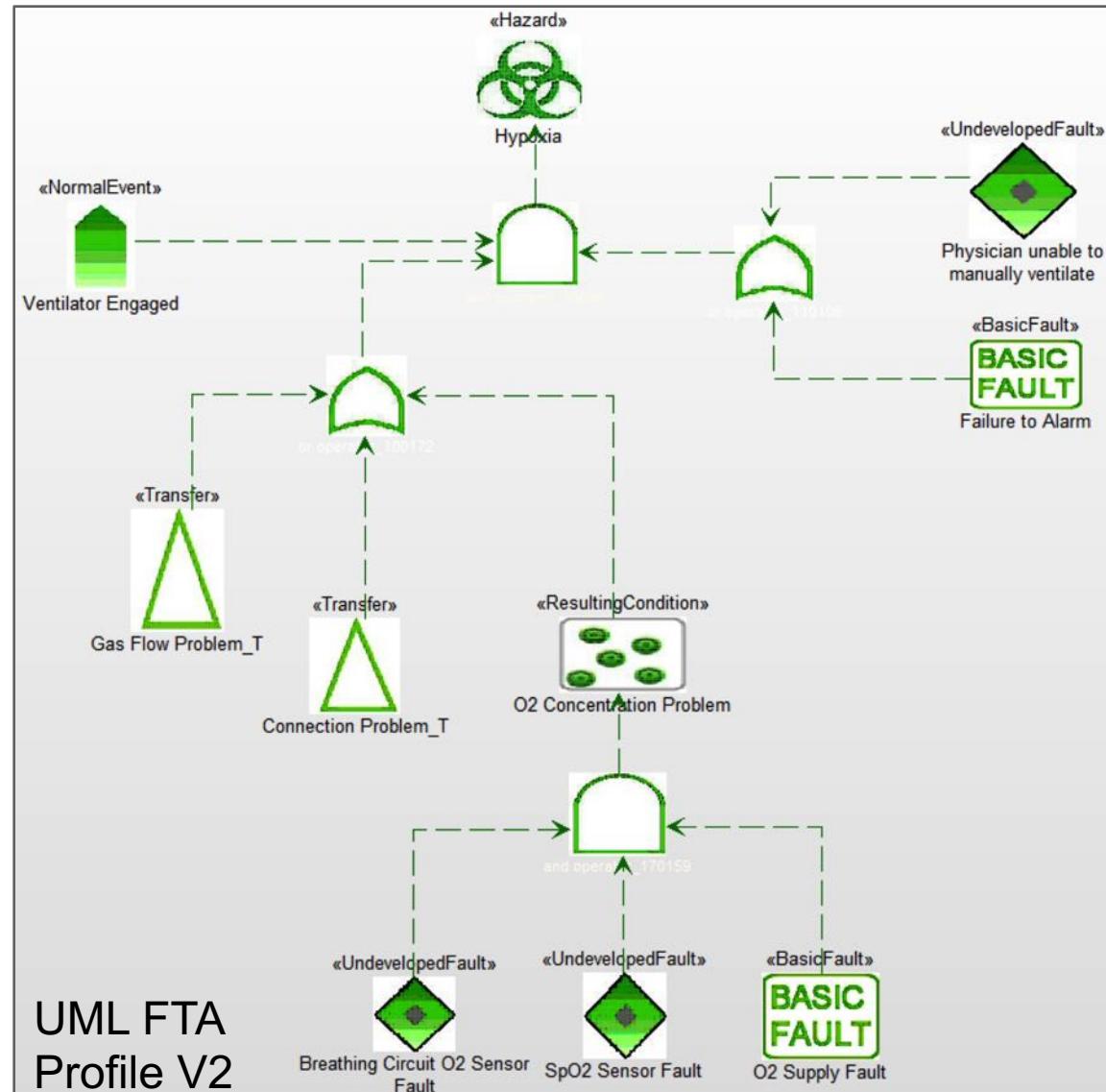
# Test-Driven Development for MBSE Work Products

- The principle behind TDD is to develop and apply test cases as you develop a system to demonstrate that it is correct
  - This is done in parallel with the system development and not ex post facto
  - This is about defect avoidance not so much defect identification and repair
- TDD applies to the development of complex system use case, architecture and design models

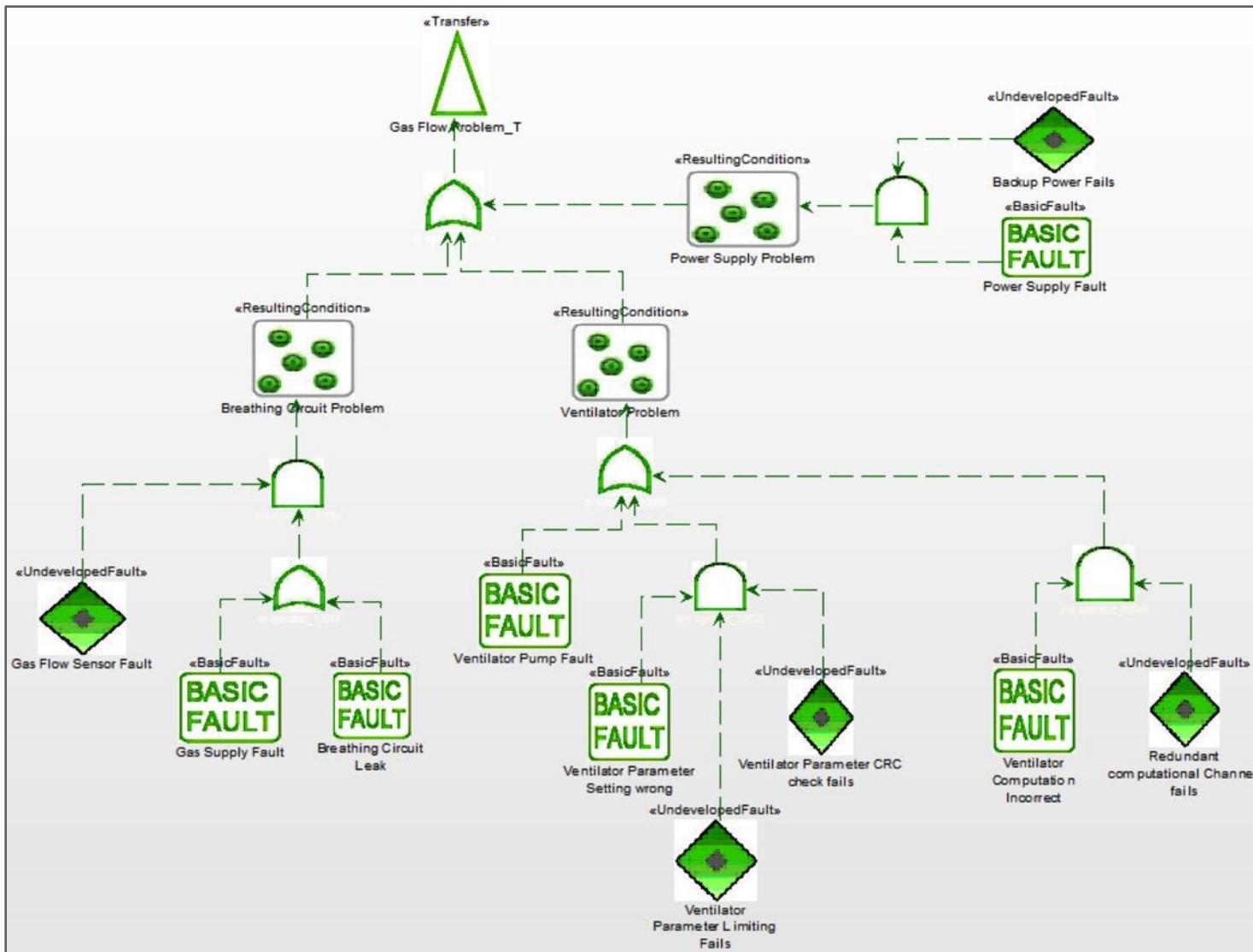


# Integrated Safety and Reliability Analysis

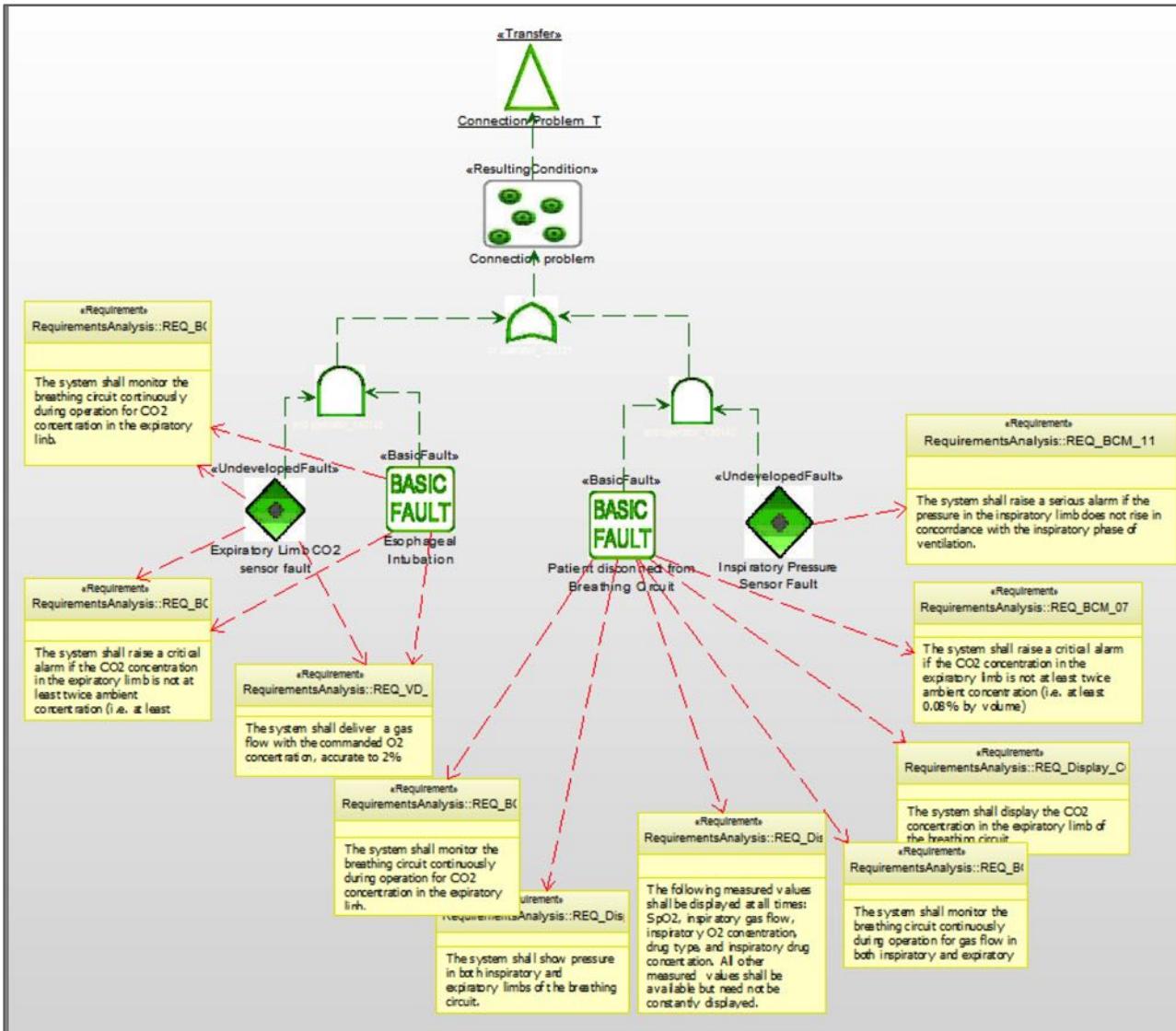
- Fault Tree Analysis (FTA) connects *hazards* with logical combinations of events, conditions, errors, and faults
- Allows you to identify
  - ▶ Effects of combinations of conditions and events on safety
  - ▶ Safety measures
  - ▶ Safety requirements
  - ▶ Impacts of architectural, technological, and design choices on safety



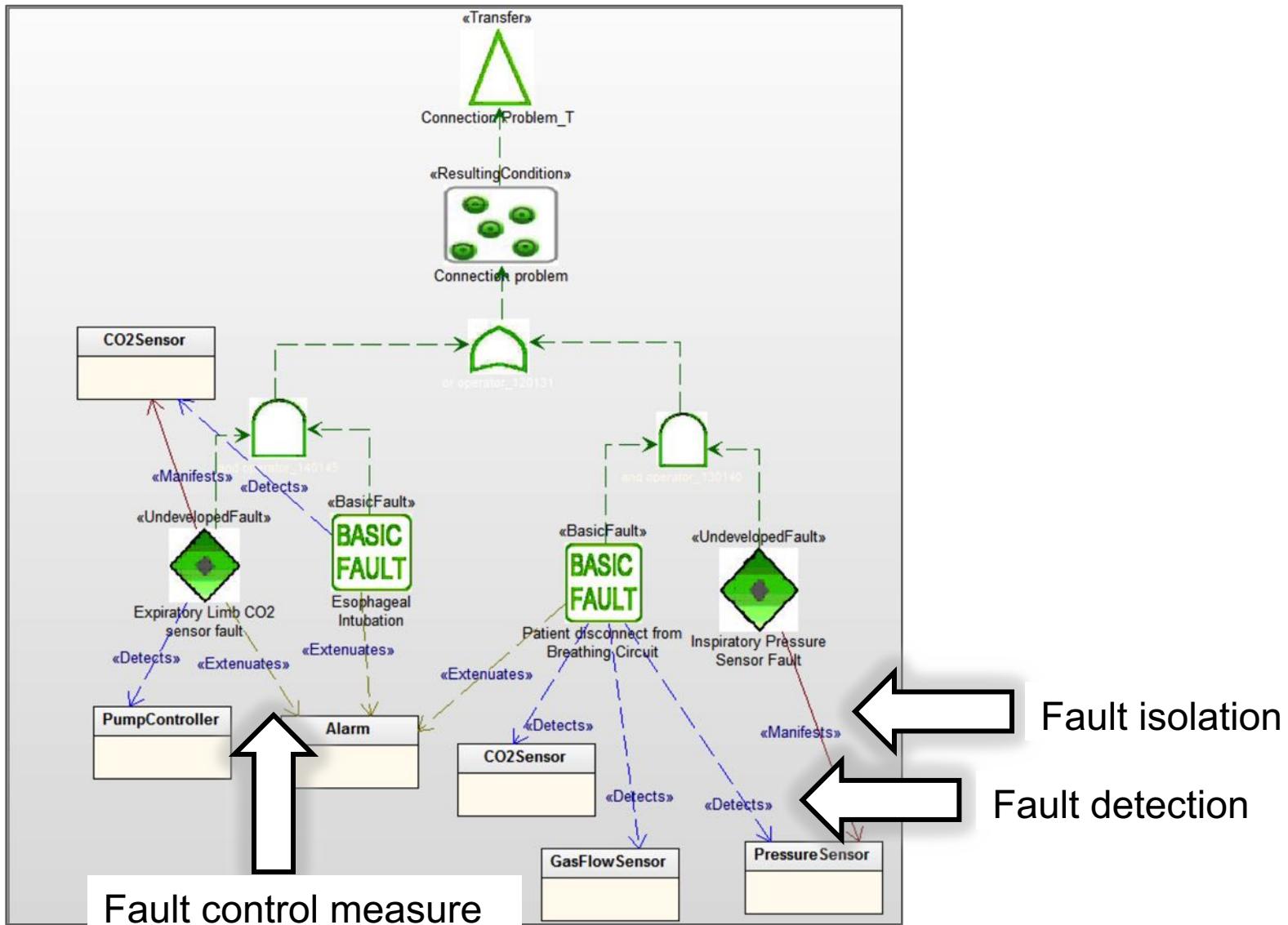
# Integrated Safety and Reliability Analysis



# Mapping Requirements to Fault Tree Analysis



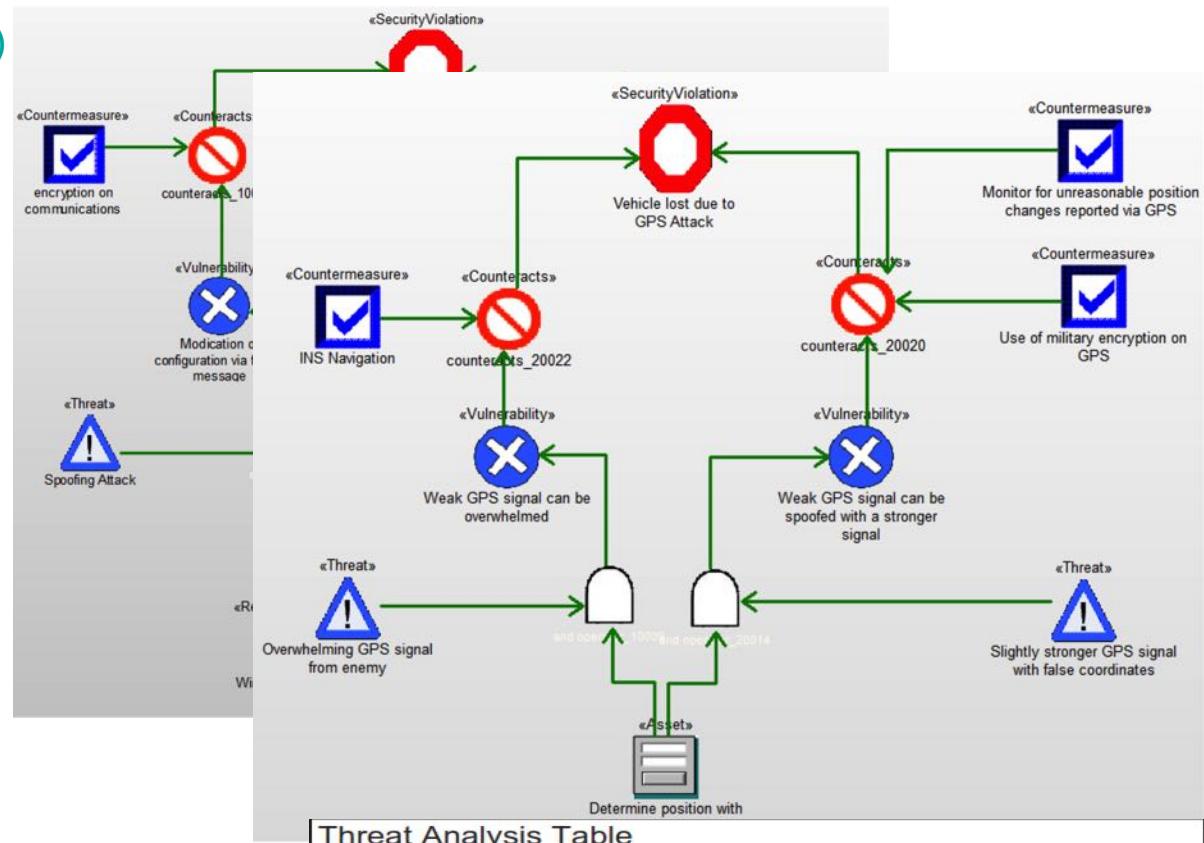
# Mapping Design to Fault Tree Analysis



# Model-Based Threat Analysis

## ■ Security Analysis Diagram (SAD) is like a Fault Tree Analysis (FTA) but for security, rather than safety

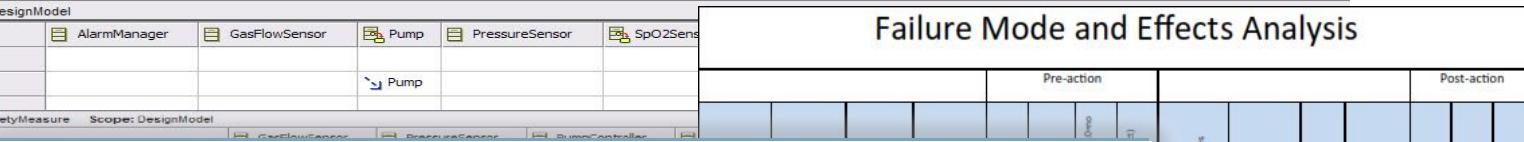
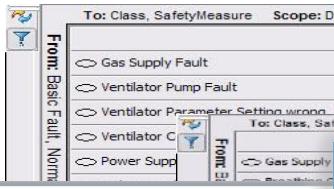
- It looks for the logical relation between assets, vulnerabilities, attacks, and security violations
- Permits reasoning about security
  - What kind?
  - How much?
  - Risk assessments
  - Cost of security penetration
  - Adequacy of countermeasures
  - Who has access to assets



Threat Analysis Table

Threat Analysis Table										
Asset value is the value of the asset to be protected (1=very low, 10=very high).										
Likelihood is the probability of the attack (1=very low, 10=certain).										
Reproducibility refers to how easy it is to reproduce the attack (for example, does it depend on timing or other circumstances?) (1=hard, 10= very easy).										
Exploitability refers to how easy it is to launch the attack (1=very easy, 10=very hard).										
Breadth is the a measure of the extent of the attack. How widespread is it or how many systems are affected? (1=few, 10=very many).										
Discoverability is how easy is it for outsiders to find out about and exploit the vulnerability (1=very easy, 10=very hard).										
Threat Priority is the product of the above values and is used to prioritize the threats for countermeasures.										
These are in the range of 1 - 10										
Asset	Vulnerability	Threat Vector	Asset Value	Likelihood of attack	Reproducibility	Exploitability	Breadth	Discoverability	Threat Priority	Countermeasure
Patient Demographic Data	Access via Ethernet	Input validation weak	4	7	9	4	1	9	9072	Internal encryption
	Access via USB	Auto-execution of USB SW	4	7	9	3	1	9	6504	Internal encryption
	Access via packet snoping	Messages sent in plain text	4	7	9	5	1	6	10080	Message encryption

# Auto-generation of summary documentation from models



Documents are generated automatically from engineering work in models

**INTERFACE CONTROL**  
Source: Model A76-BrakingSubsystemModel v2.1  
Subsystem: Braking Management Subsystem

Interface: iBrakingForce  
Service: GetBrakingForce  
Data: brakingForce  
Type: Scaled 32-bit integer  
Media: CAN Bus message  
Range: 0..1000  
Accuracy: ± 0.05 N  
Return value: none  
Rate: 5 ms  
Worst Case Response time: 1 ms

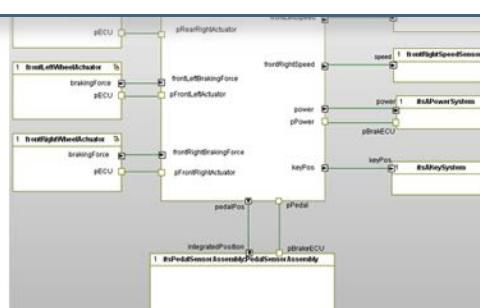
Interface: iBrakingCommands  
Service: EnableBrakingAugmentation  
Data: enable  
Type: 1-bit  
Media: CAN Bus message  
Range: 0 (FALSE) .. TRUE (1)  
Accuracy: N/A  
Return value: ACK\_Type  
Rate: no faster than 2/s  
Worst Case Response Time: 2 ms

Service: PowerOnSelfTest  
Data: None  
Return value: POST\_Return\_Type  
Rate: No faster than 1/10 minutes  
Worst Case Response Time: 1000 ms

Service: CalibrateBrake

## Threat Analysis Table

Asset value is the value of the asset to be protected (1=very low, 10=very high).					
Likelihood is the probability of the attack (1=very low, 10=certain).					
Reproducibility refers to how easy it is to reproduce the attack (for example, does it depend on timing or other conditions).					
Exploitability refers to how easy it is to launch the attack (1=very easy, 10=very hard).					
Breadth is a measure of the extent of the attack. How widespread is it or how many systems are affected?					
Discoverability is how easy is it for outsiders to find out about and exploit the vulnerability (1=very easy, 10=very hard).					
Threat Priority is the product of the above values and is used to prioritize the threats for countermeasures.					
These are the threat vectors:					
Asset	Vulnerability	Threat Vector	Asset Value	Likelihood of attack	Reproducibility
Patient Demographic Data	Access via Ethernet	Input validation weak	4	7	9
	Access via USB	Auto-execution of USB SW	4	7	9
	Access via socket snoping	Messages sent in plain text	4	7	9
			5		



Failure Mode	Severity	Probability	Risk	Safety Integrity Level
Overpressure can damage the lungs. This is an especially severe trauma, possibly fatal, to neonates.	200 milliseconds	1.00E+04	4	3.00E+04
Hyperoxia problems are usually limited to neonates, where it can cause blindness.	10 minutes	1.00E+05	4	4.00E+05
Inadequate anesthesia leads to patient discomfort and memory retention of the surgical procedures. This is normally not life threatening but can be severely discomforting.	5 minutes	1.00E+04	2	2.00E+04
Over anesthesia can lead to death.	3 minutes	1.00E+03	4	4.00E+03
Anesthesia leak can lead to short or, in smaller doses, to long-term poisoning of medical staff.	10 minutes	1.00E+05	5	4.00E+05

# Harmony Agile MBSE Delivery Process

Rational Method Composer

Team (IBM)

Welcome to the Rational Harmony Agile Model-Based Systems Engineering

Welcome to the Rational Harmony Agile Model-Based Systems Engineering

The Rational Harmony Agile Model-Based Systems Engineering (aMBSE) process is a delivery process for the development of systems engineering data and work product using both model-based systems techniques with UML and SysML but is at the same time agile and incorporates agile practices for improved quality and engineering efficiency.

Main Description

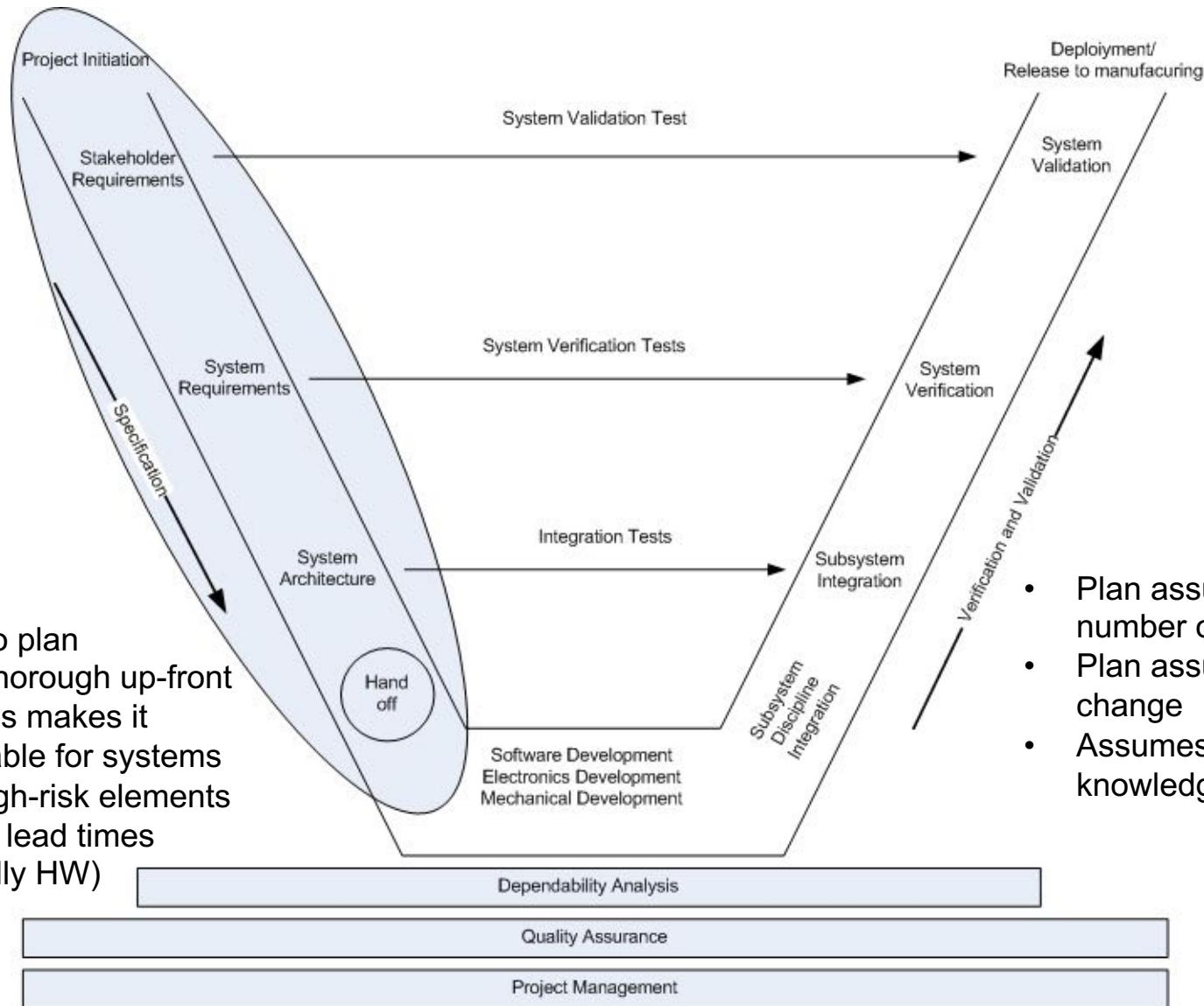
**Harmony aMBSE**  
Agile Model-Based Systems Engineering

With the initial release of the UML in 1995, systems engineers had a standard language in which they could express requirements, architectures, designs, and other kinds of engineering data. However, there was widespread belief that the Unified Modeling Language (UML) itself was too "software oriented" for general use in systems engineering which led to the development and release of the Systems Modeling Language (SysML). UML and SysML provide a number of key advantages for the development of system engineering data:

- Precision of engineering data
- Data consistency across work products and engineering activities
- A common source for engineering truth
- Improved visualization and comprehension of engineering data
- Ease of integration of disparate engineering data
- Improved management and maintenance of engineering data

和

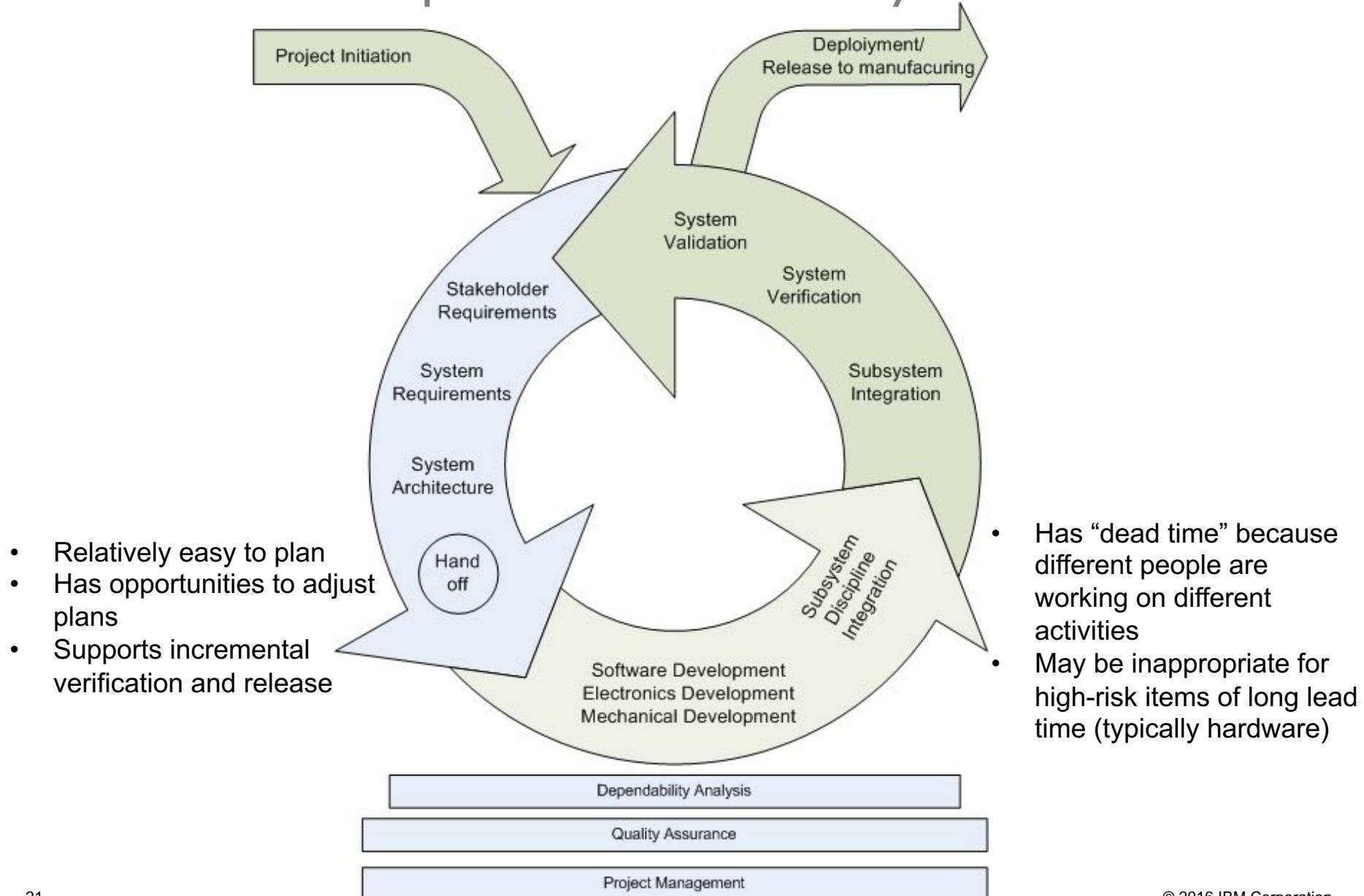
# Overall Product Development Workflows: the standard V



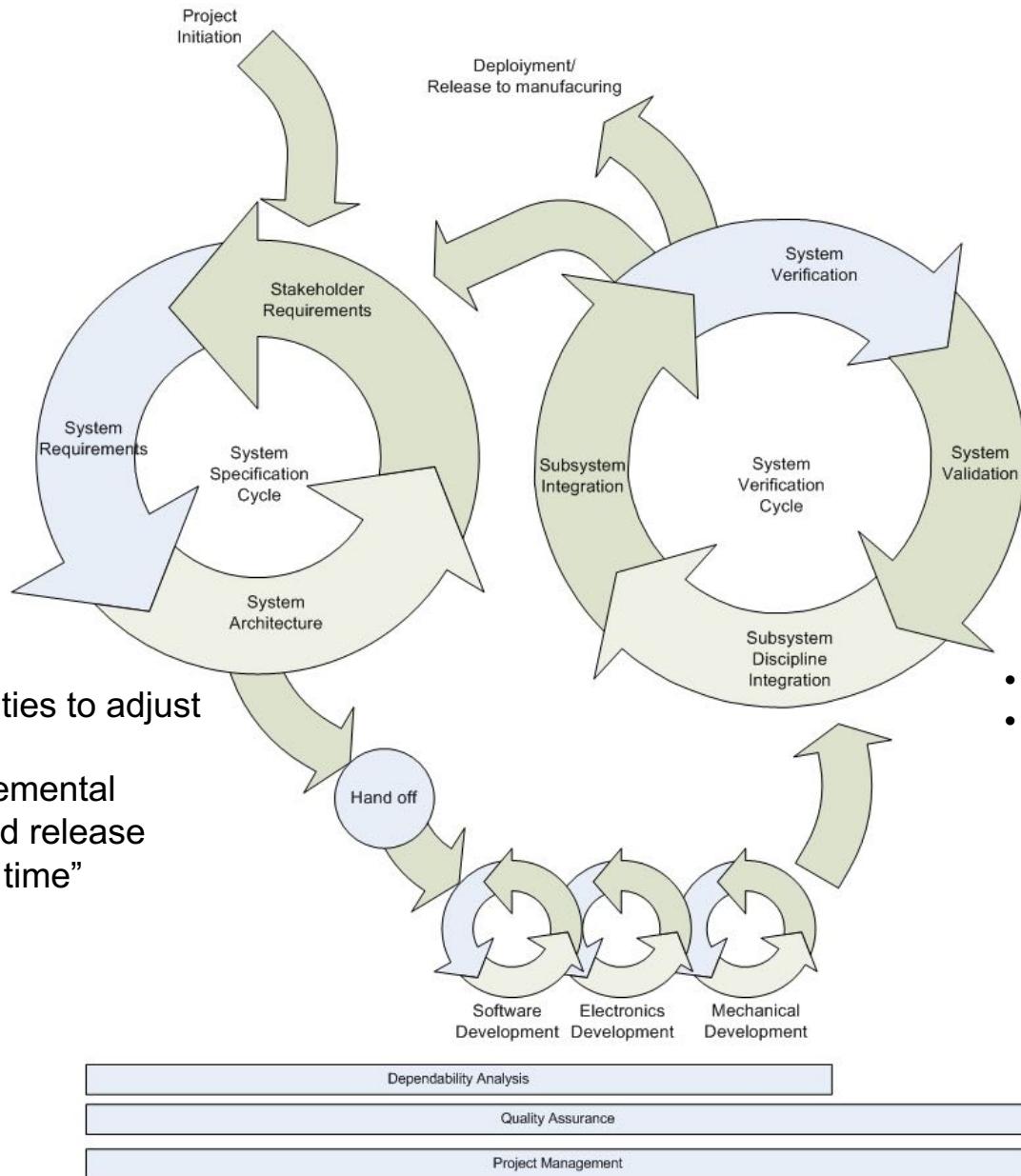
- Easy to plan
- More thorough up-front analysis makes it applicable for systems with high-risk elements of long lead times (typically HW)

- Plan assumes small number of minor defects
- Plan assumes little or no change
- Assumes “infinite knowledge”

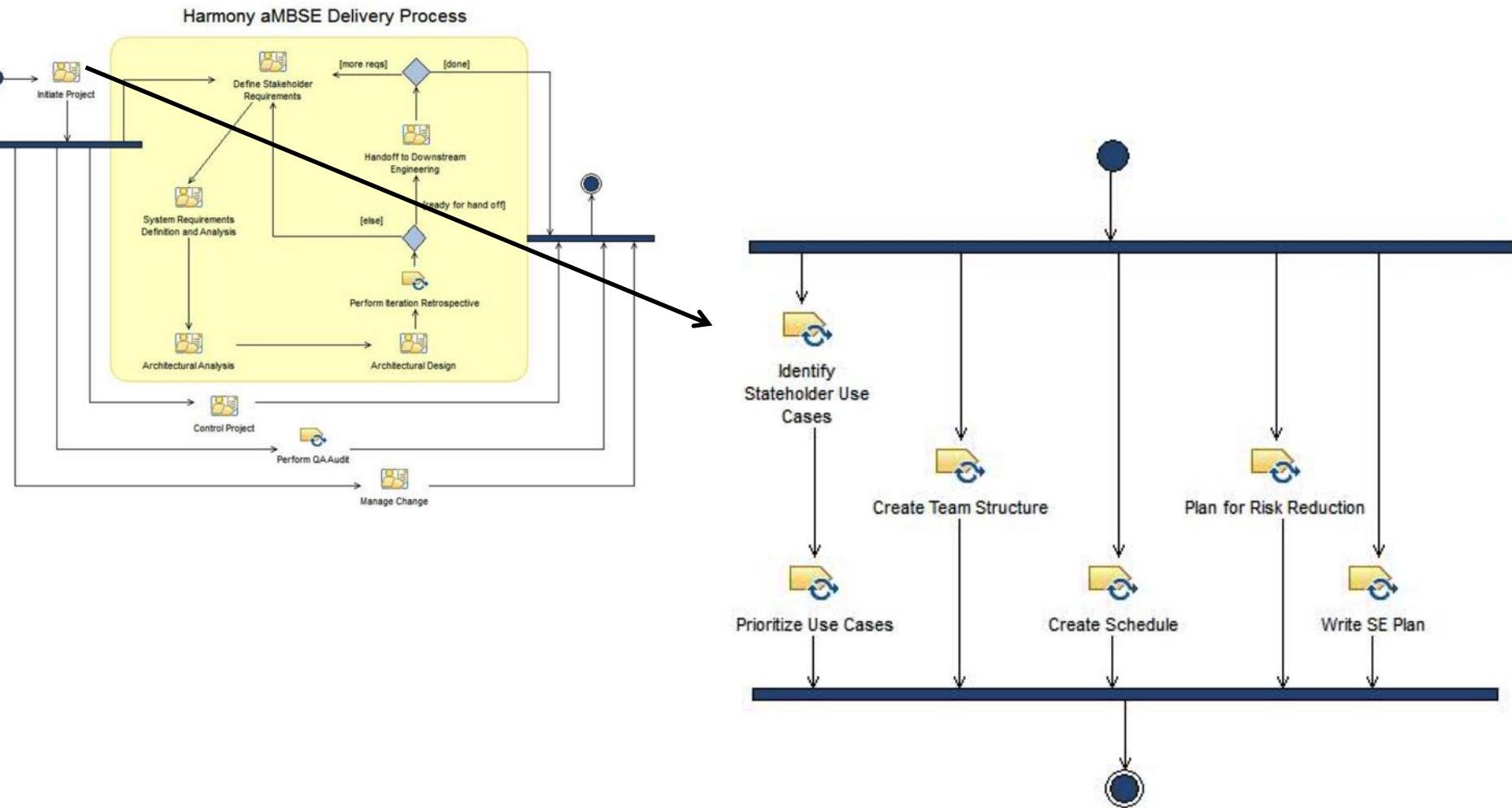
# Overall Product Development Workflows: the fully incremental



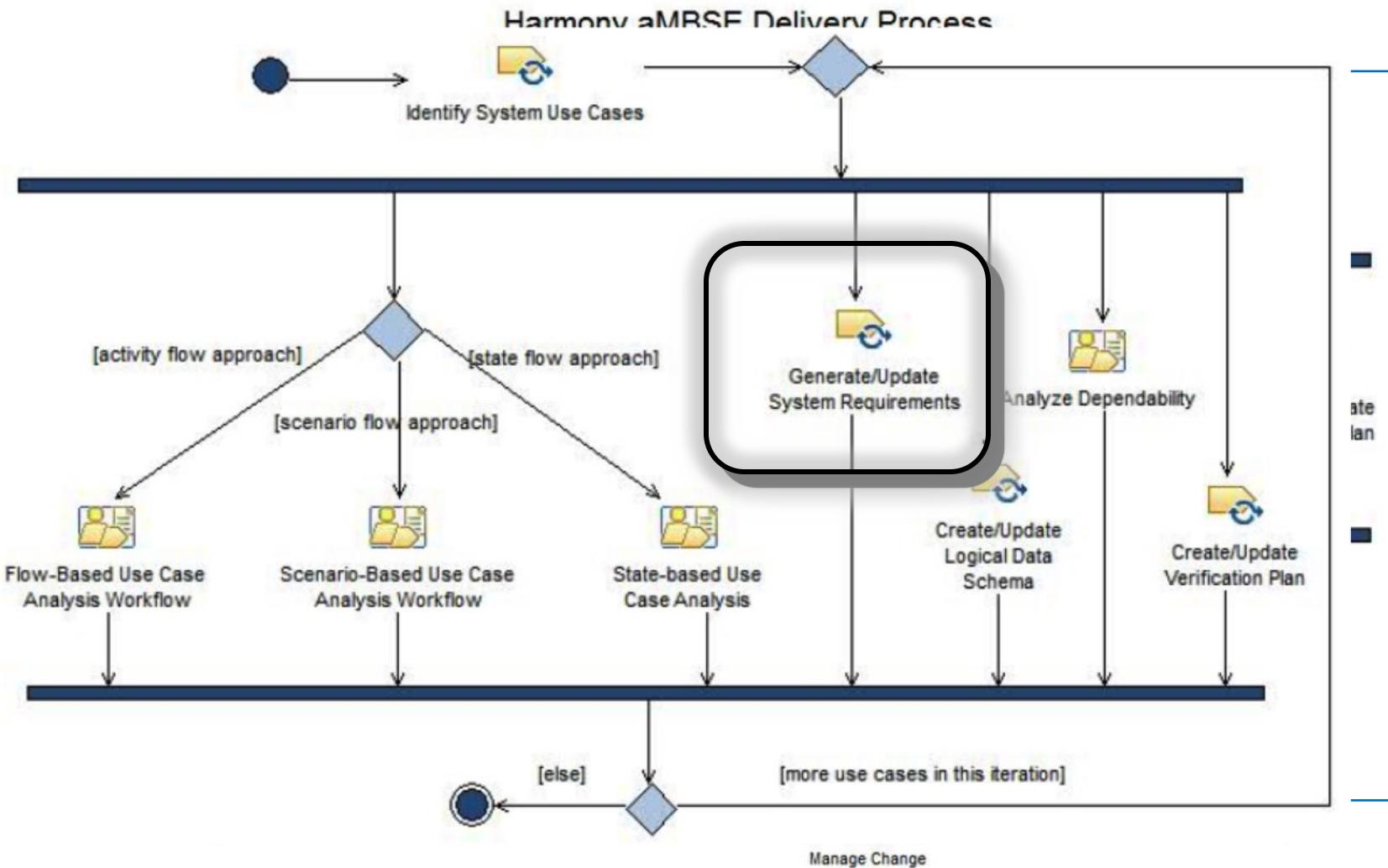
# Overall Product Development Workflows: the hybrid agile V



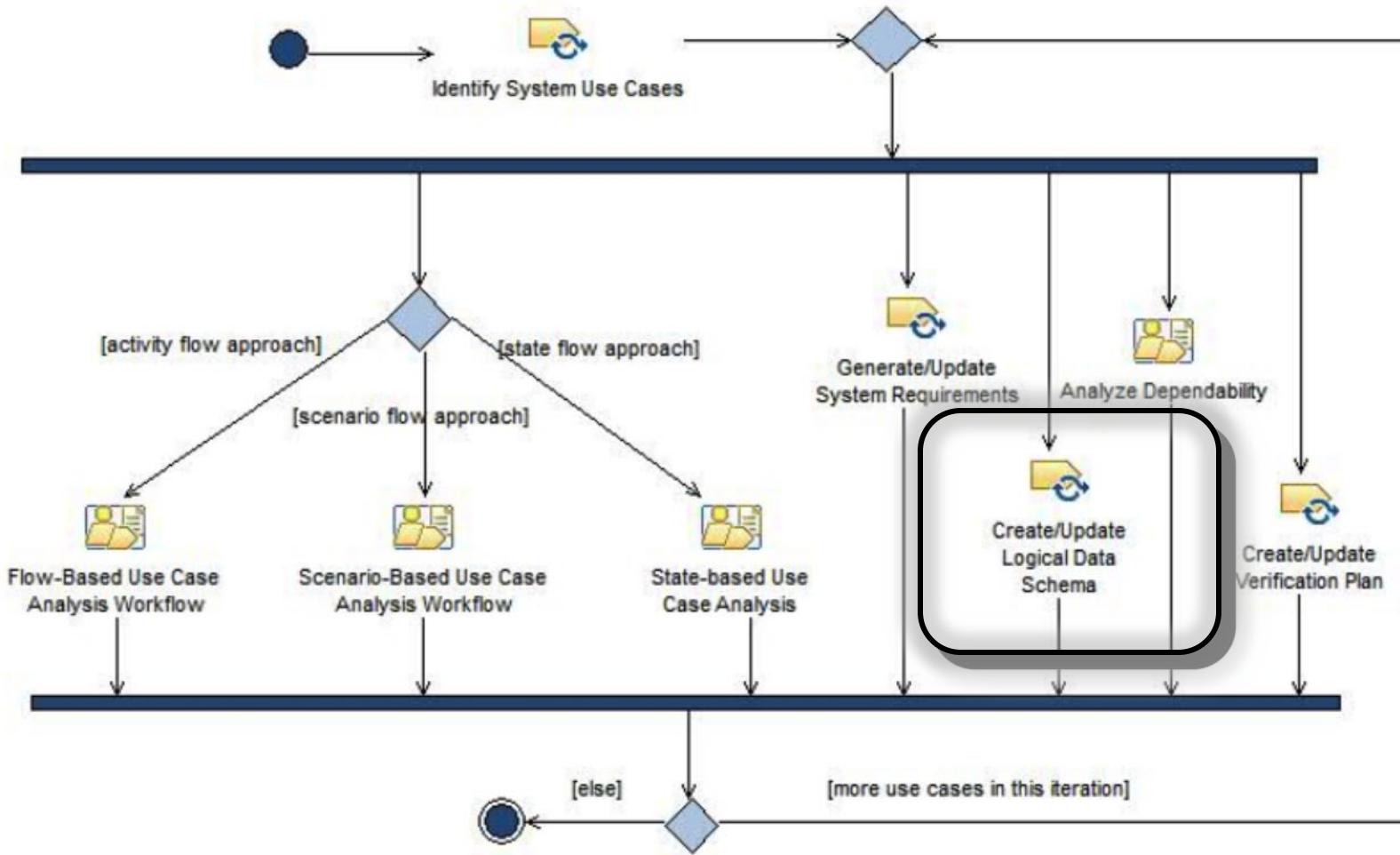
# Initiate project



# Harmony Process for Agile MBSE

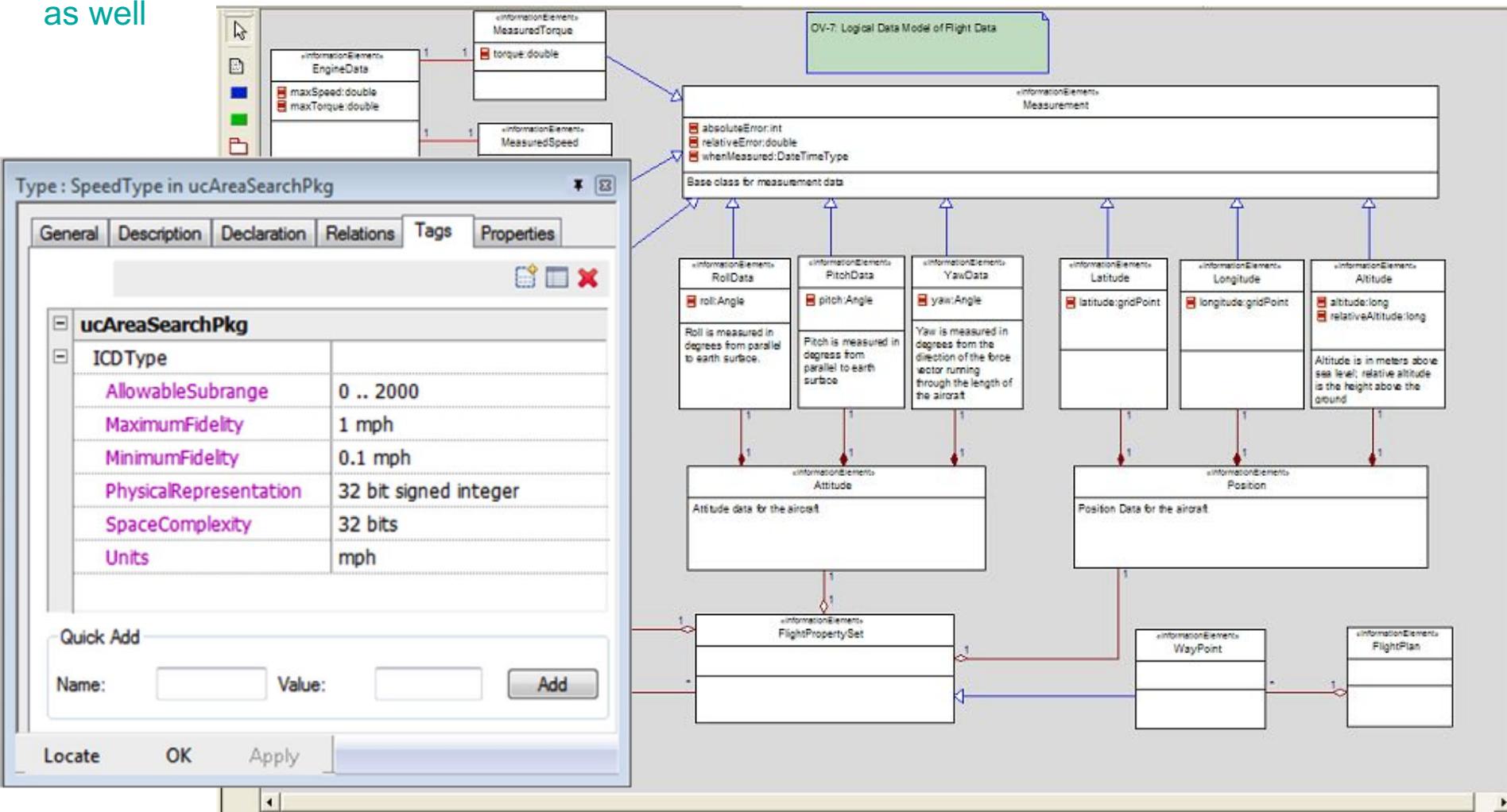


# Harmony Process for Agile MBSE

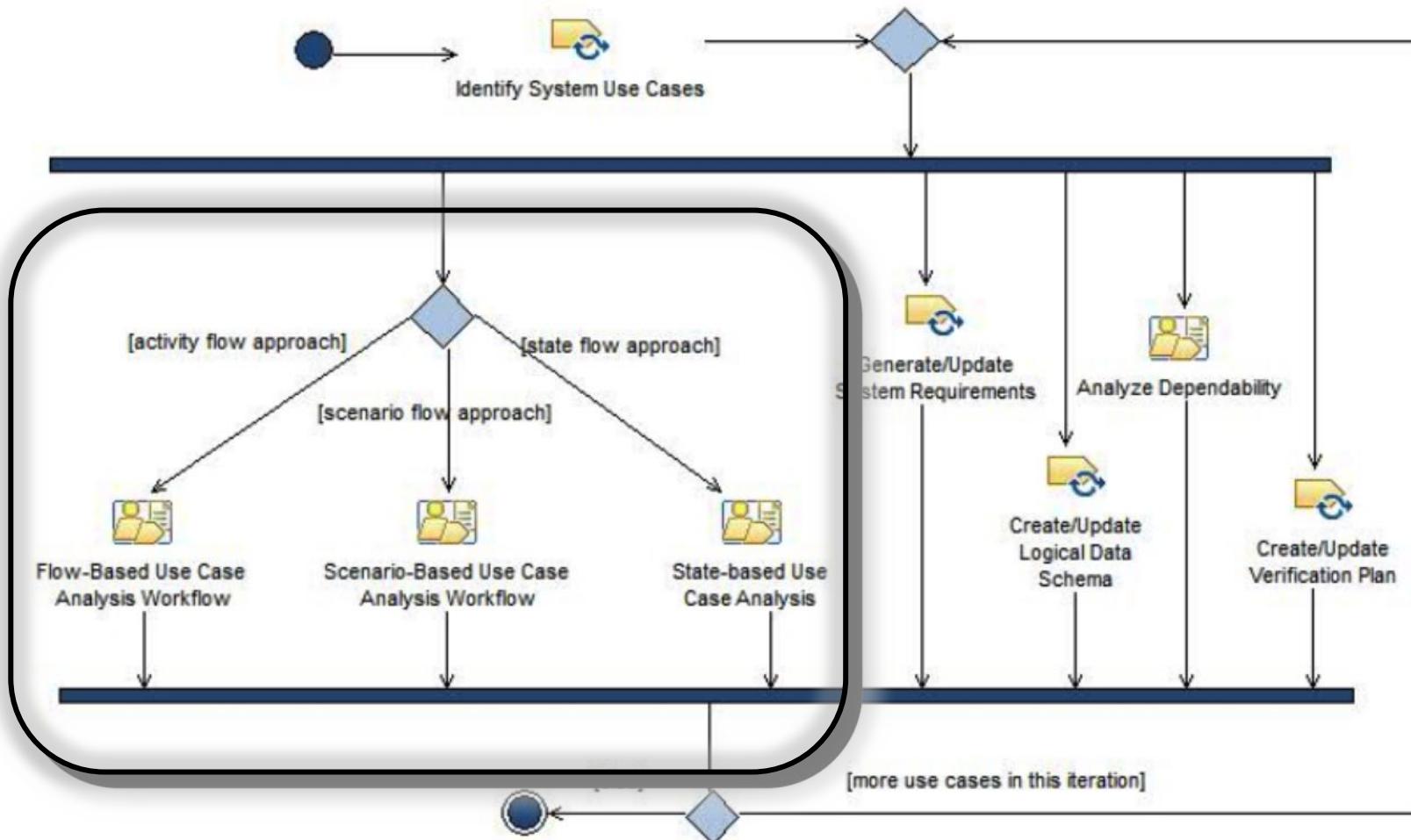


# Logical Data Schema Modeling

- A logical data schema identifies the logical properties of important data elements and types and the relations among such data elements and their metadata
- Although the name is “data schema” it includes physical, materiel, and energy flows specification as well



# Harmony aMBSE: System Requirements Def & Analysis



# Alternative approaches to Build Executable Model of UC

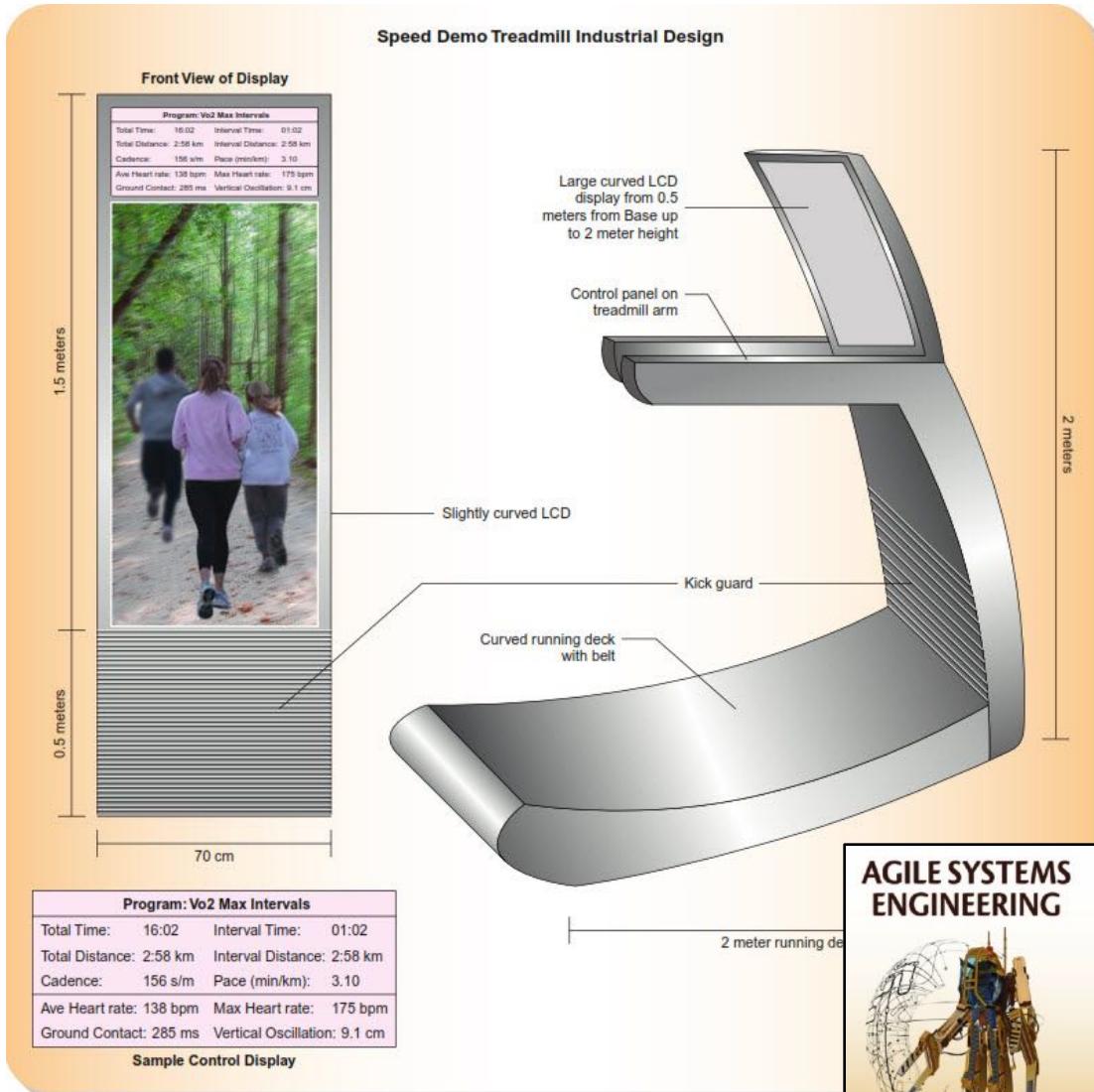
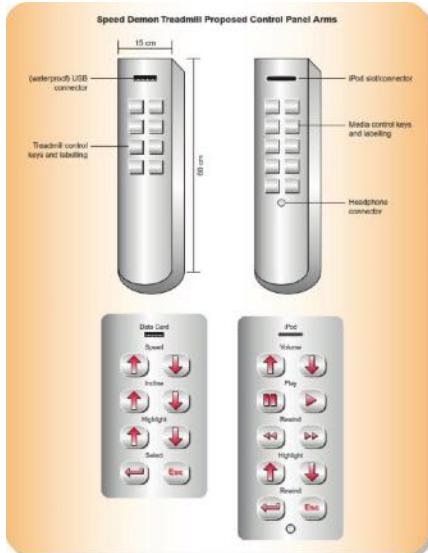
Alternative 1: Scenario Based		Alternative 2: Flow Based		Alternative 3 State Based	
Create model context	<i>Block Diagram</i>	Identify functional flow large-scale view	<i>Activity diagram</i>	Create Model Context	<i>Block Diagram</i>
Identify sequences of messages between system use case and actors	<i>Sequence Diagram</i>	Derive sequences from functional flow	<i>Sequence diagram</i>	Create executable state machine	<i>State Diagram</i>
(optional) cluster sequences together as flows	<i>Activity Diagram</i>	Create model context	<i>Block Diagram</i>	Identify interfaces	<i>Block Diagram</i>
Identify interfaces	<i>Block Diagram</i>	Identify interfaces	<i>Block Diagram</i>	Derive sequences from functional flow	<i>Sequence diagram</i>
Create executable state machine	<i>State Diagram</i>	Create executable state machine	<i>State Diagram</i>	Execute State machine	<i>Model execution views</i>
Execute State machine	<i>Model execution views</i>	Execute State machine	<i>Model execution views</i>	Repeat until all requirements and sequence variants covered	
Repeat until all requirements and sequence variants covered		Repeat until all requirements and sequence variants covered			

# Today's Example: High Performance Treadmill: SpeedDemon™

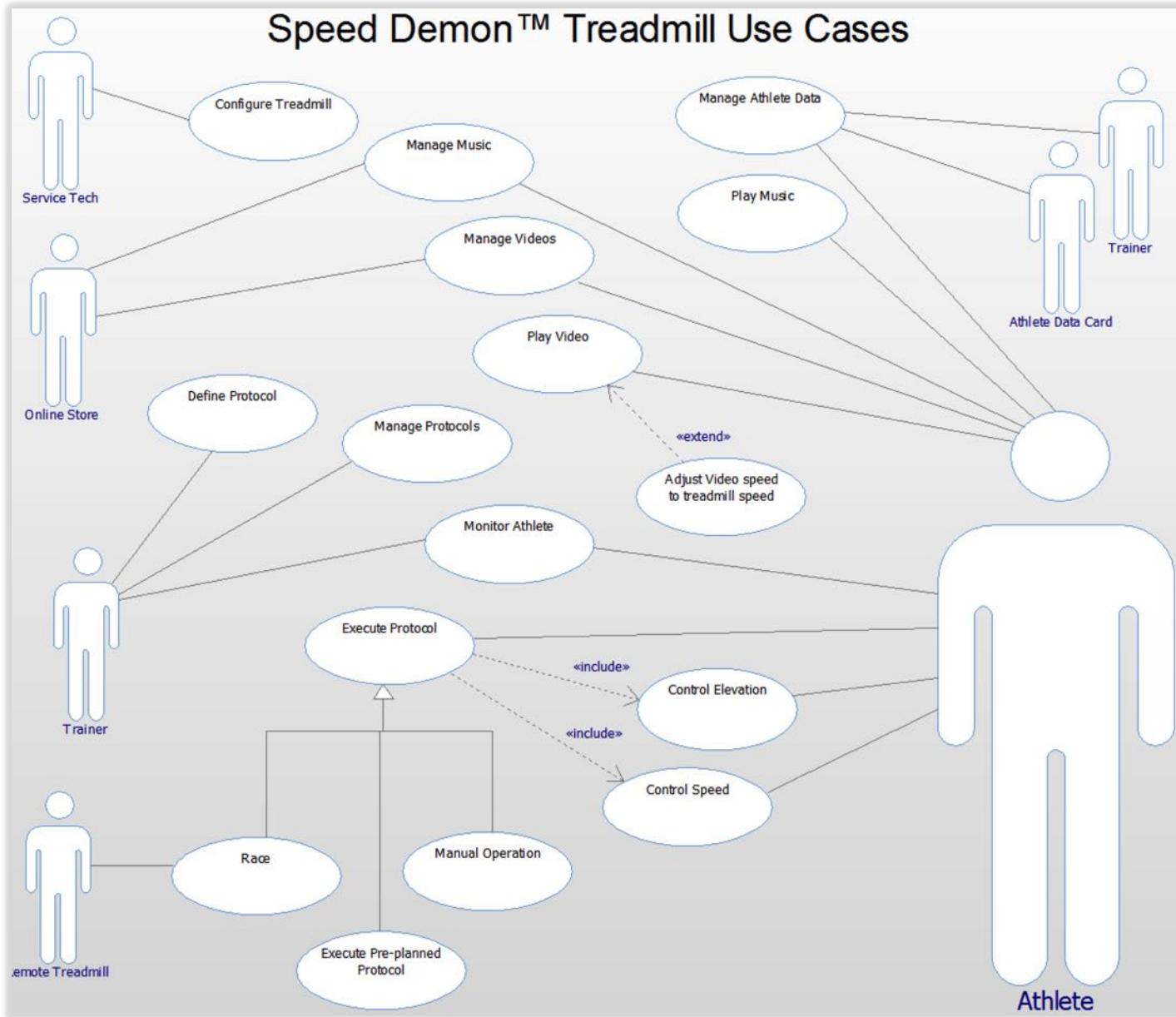
## ▪ Biometrics

- Heart rate
- Speed
- Power / VO<sub>2</sub> max
- Elapsed time/interval time
- Ground contact time
- Vertical oscillation
- Cadence

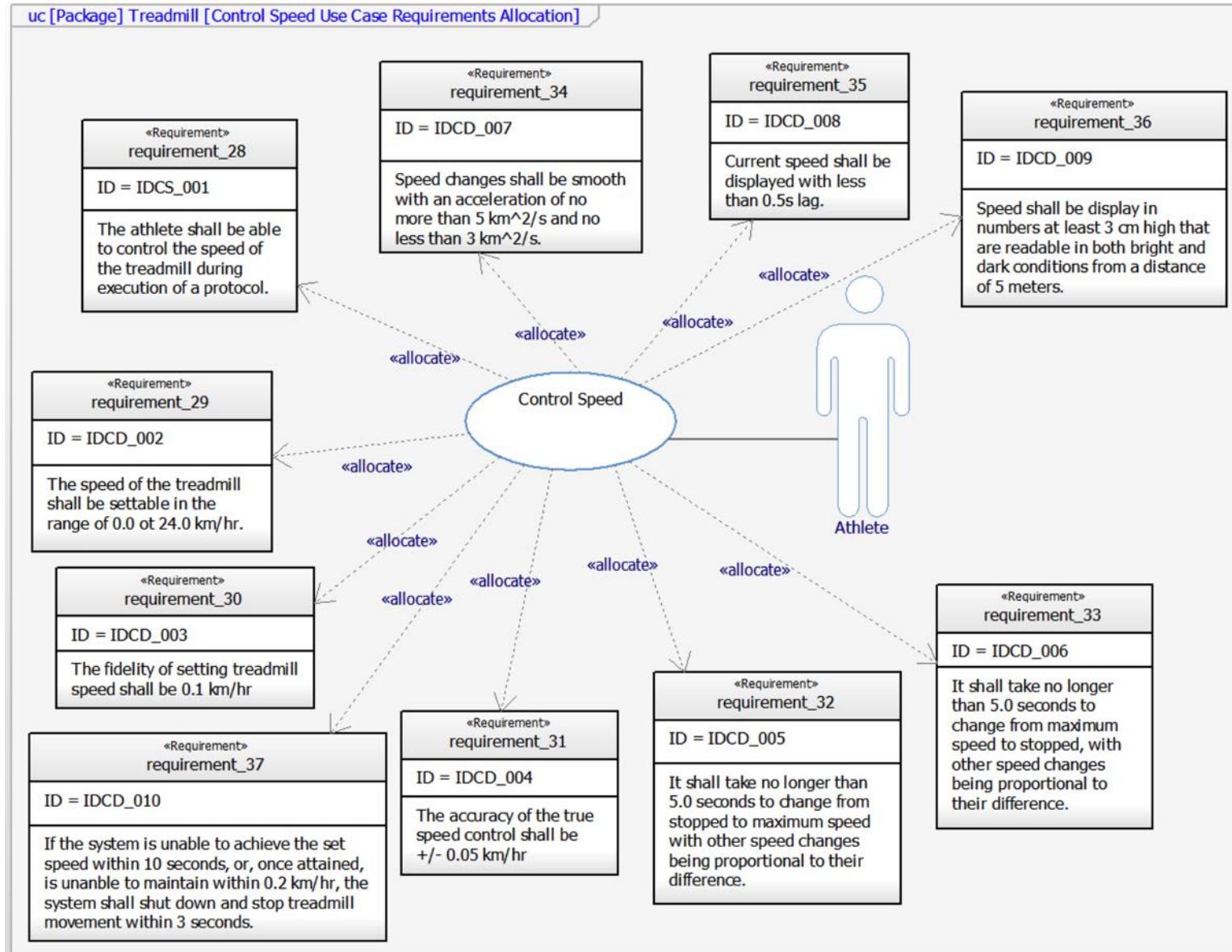
- Programmable protocols (workouts)
- Performance storage and analytics
- Music and video playlists
- Virtual racing (simulated or over-the-web)



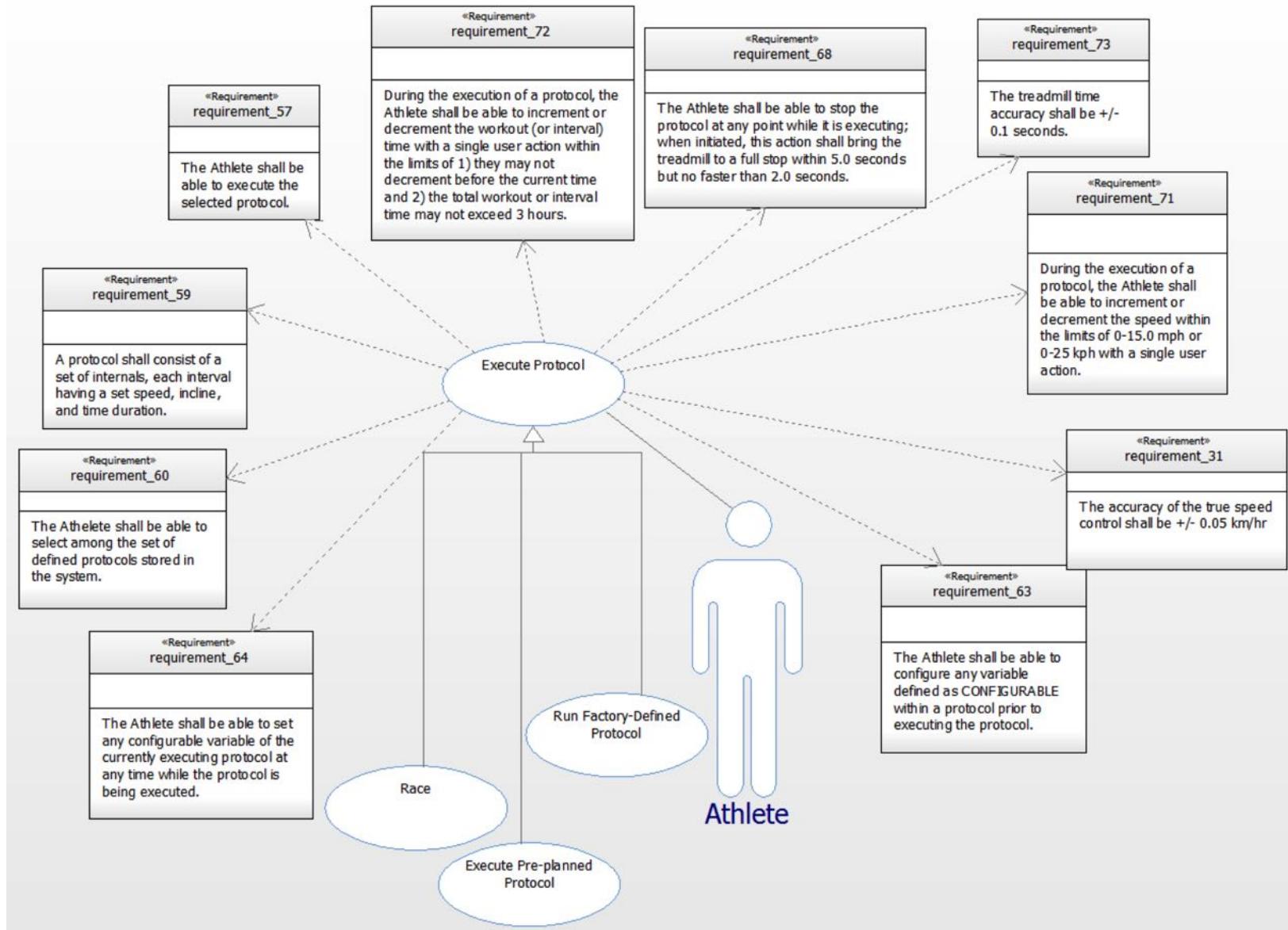
# SpeedDemon Use Cases



# Allocation of requirements to Control Speed Use Case



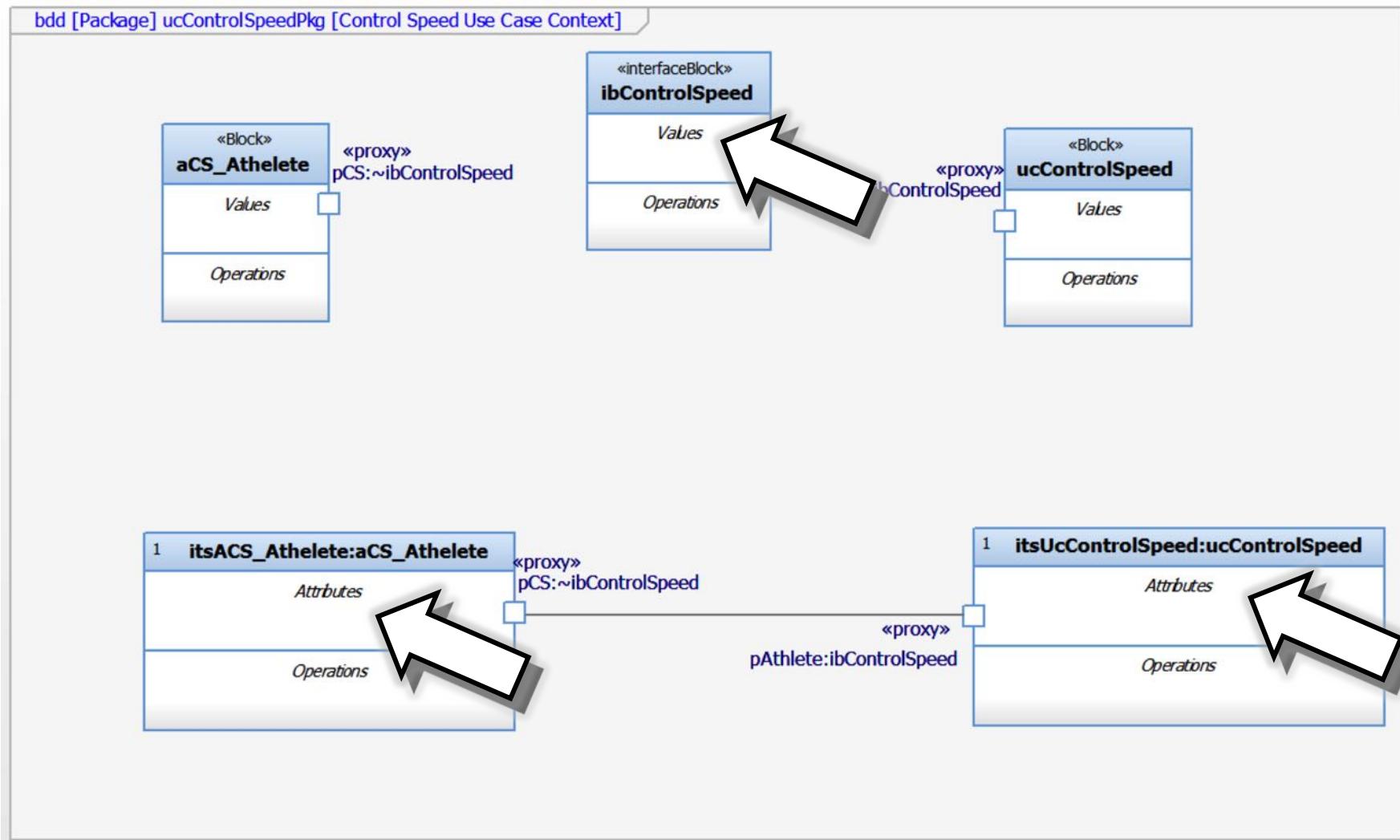
# Allocate Requirements to Execute Protocol Use Case



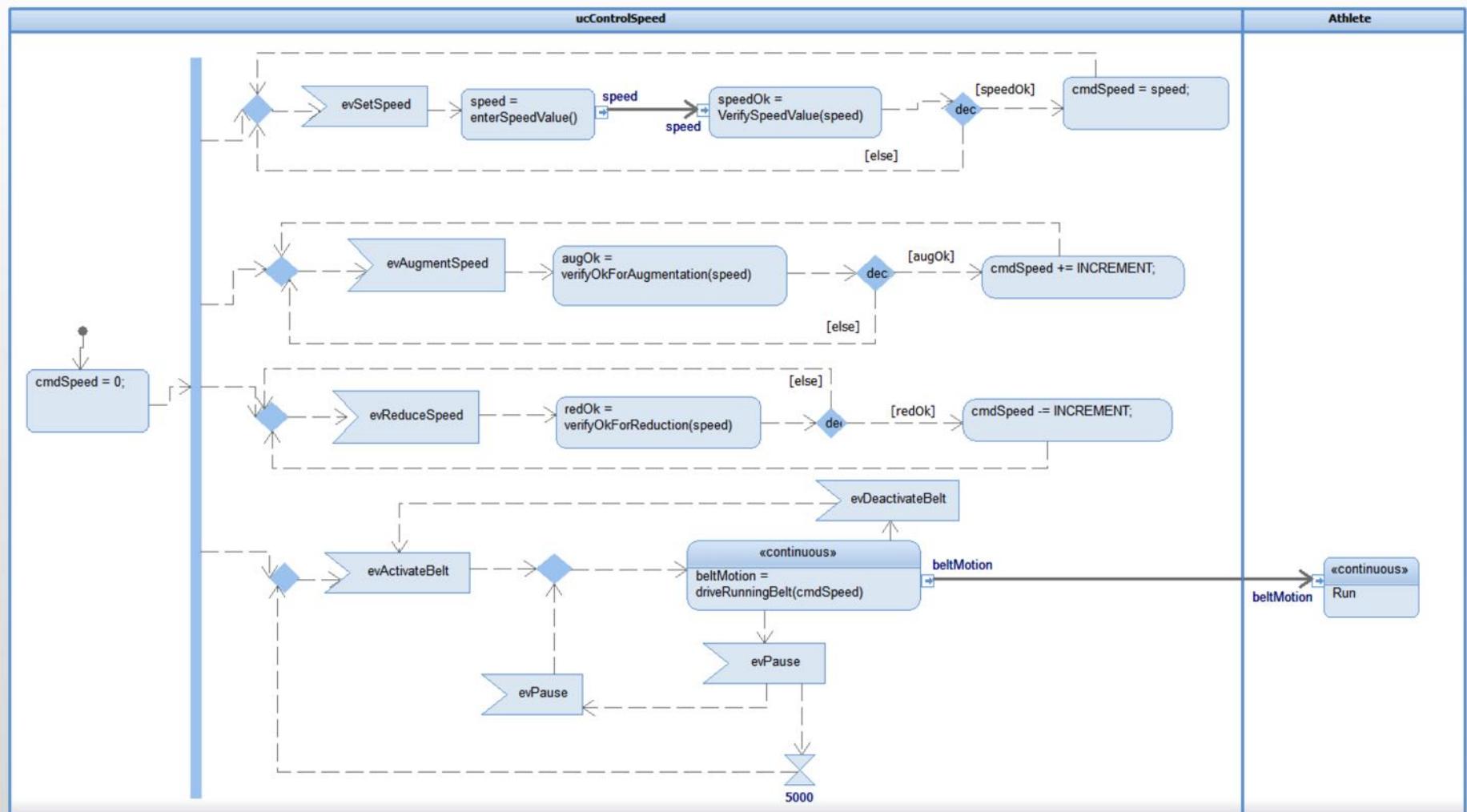
# Showing allocations within a matrix (generated)

	Control Elevation	Control Speed	Monitor Athlete	Execute Protocol	Define Protocol	Manage Protocols	Play Music	Manage Music	Play Video	Manage Videos	Run Factory-Defined Protocol	Race	Adjust Video speed to treadmill speed	Configure Treadmill	Manage Athlete Data
requirement_24															
requirement_25															
requirement_26															
requirement_28				✓ requirement_28											
requirement_29				✓ requirement_29											
requirement_30				✓ requirement_30											
requirement_31				✓ requirement_31			✓ requirement_31								
requirement_32				✓ requirement_32											
requirement_33				✓ requirement_33											
requirement_34				✓ requirement_34											
requirement_35				✓ requirement_35											
requirement_36				✓ requirement_36											
requirement_37				✓ requirement_37											
requirement_48															✓ requirement_48
requirement_49															✓ requirement_49
requirement_50															✓ requirement_50
requirement_51															✓ requirement_51
requirement_52															✓ requirement_52
requirement_53															✓ requirement_53
requirement_54															✓ requirement_54
requirement_55															✓ requirement_55
requirement_56															✓ requirement_56
requirement_57				✓ requirement_57											✓ requirement_57
requirement_59					✓ requirement_59										✓ requirement_59
requirement_60						✓ requirement_60									✓ requirement_60
requirement_61															✓ requirement_61
requirement_62															✓ requirement_62
requirement_63							✓ requirement_63								✓ requirement_63
requirement_64								✓ requirement_64							✓ requirement_64
requirement_65															✓ requirement_65
requirement_66															✓ requirement_66
requirement_67															✓ requirement_67
requirement_68								✓ requirement_68							✓ requirement_68
requirement_71									✓ requirement_71						
requirement_72									✓ requirement_72						
requirement_73										✓ requirement_73					

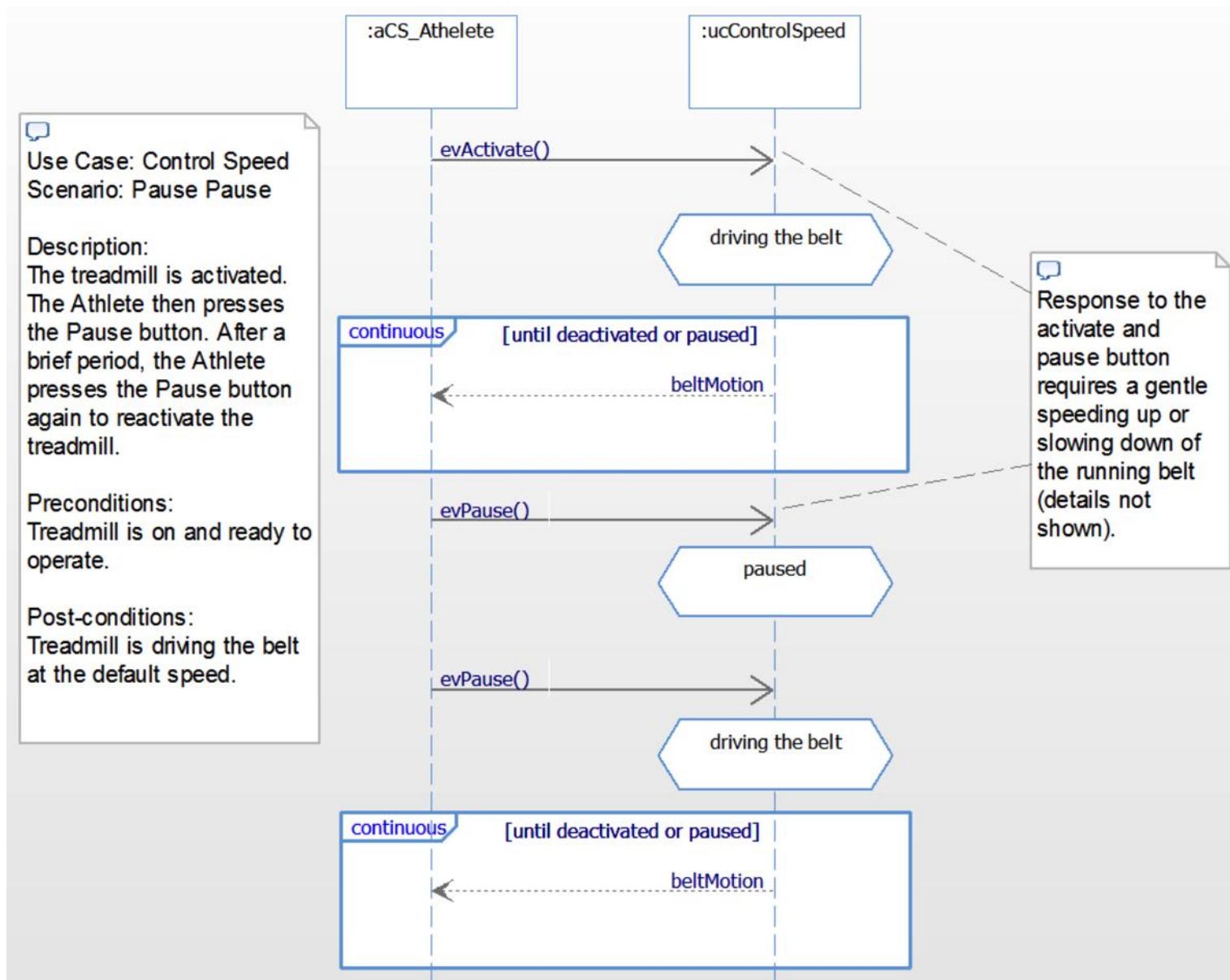
# Create context for use case execution model



## Creating an activity diagram to show behavioral flows



# Create scenarios from activity diagram



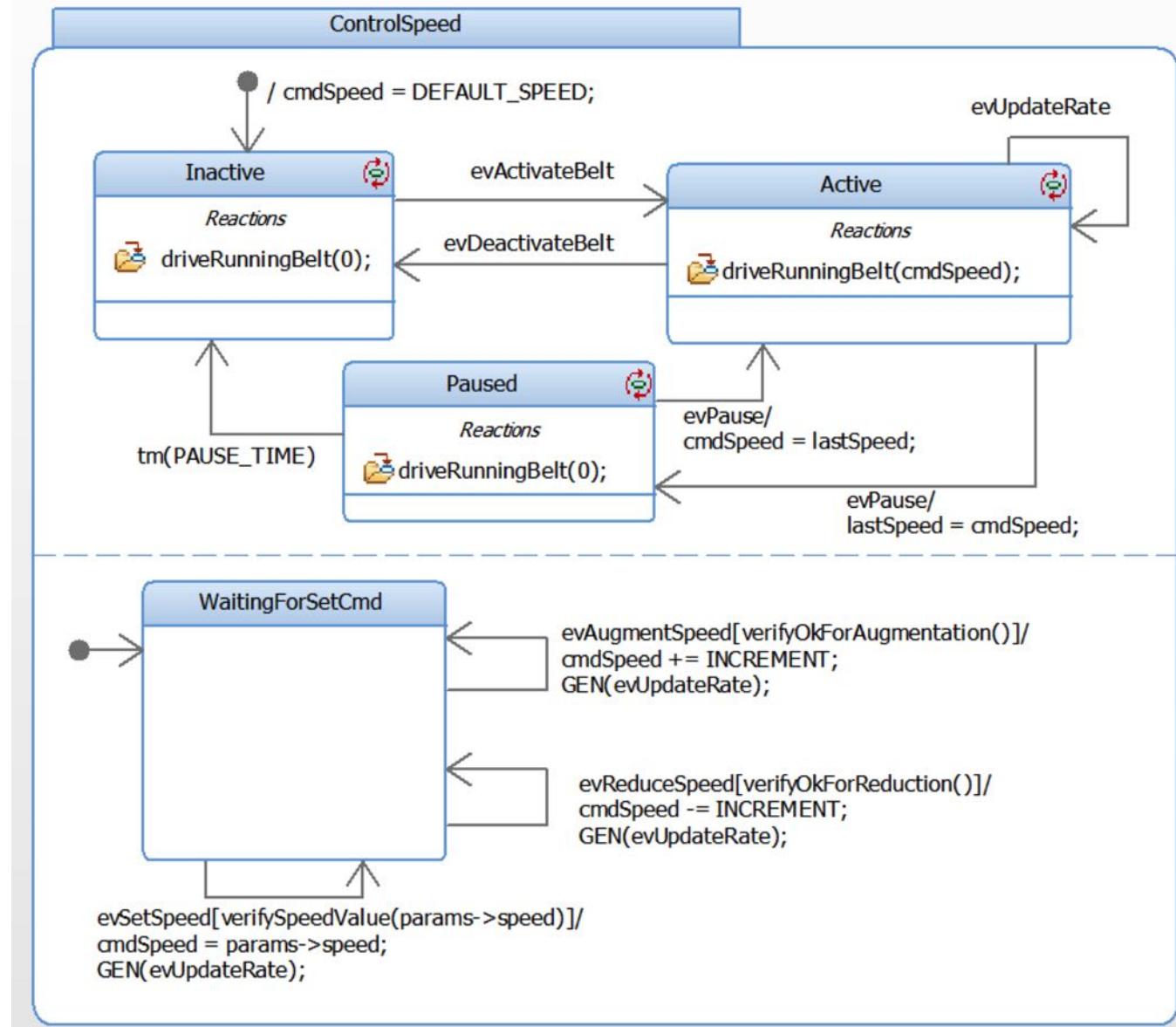
# Create Behavioral Model Realizing Requirements for Use Case

## Notes

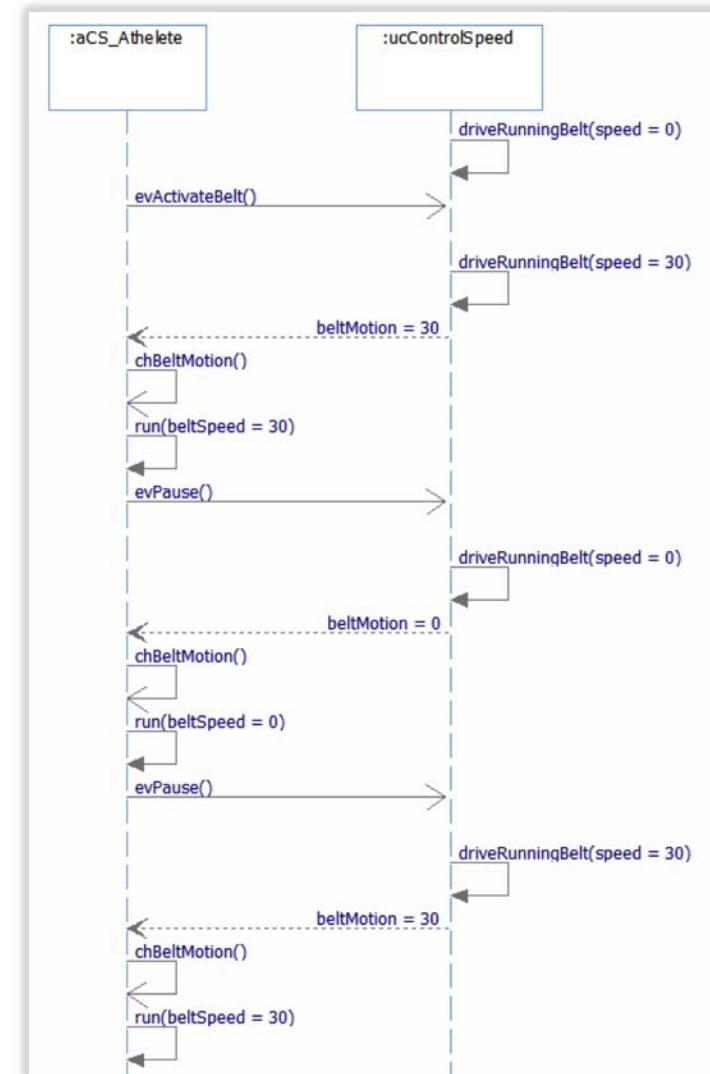
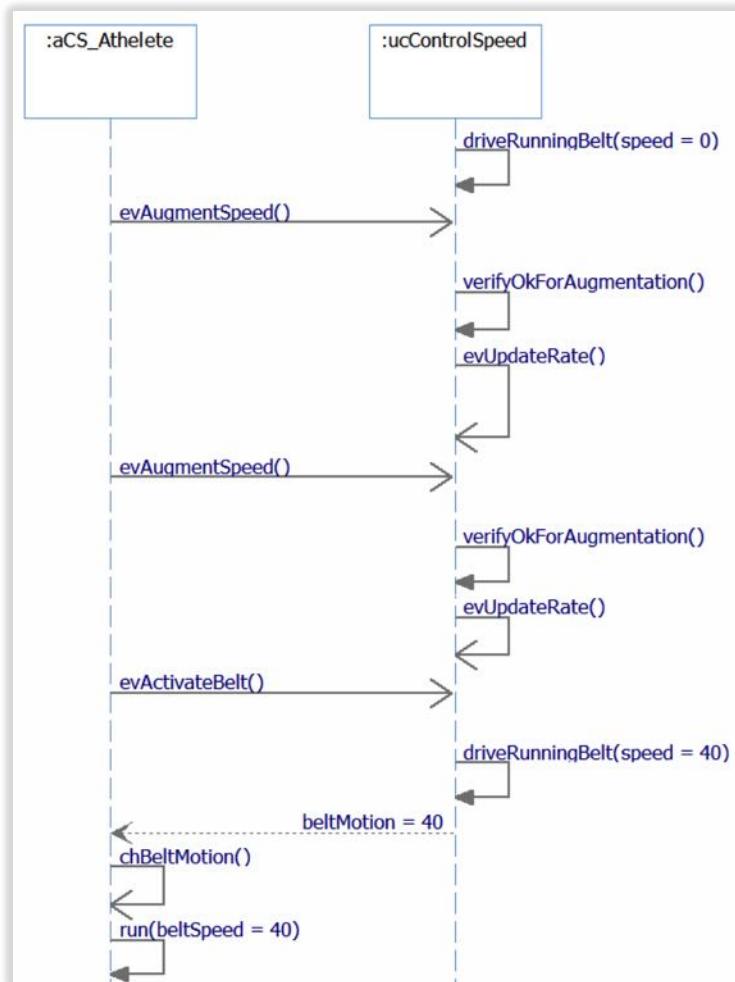
- This is after several Model-Verify– Elaborate iterations
- Start with 2-4 states, and add more each nanocycle iteration

## Questions still remain

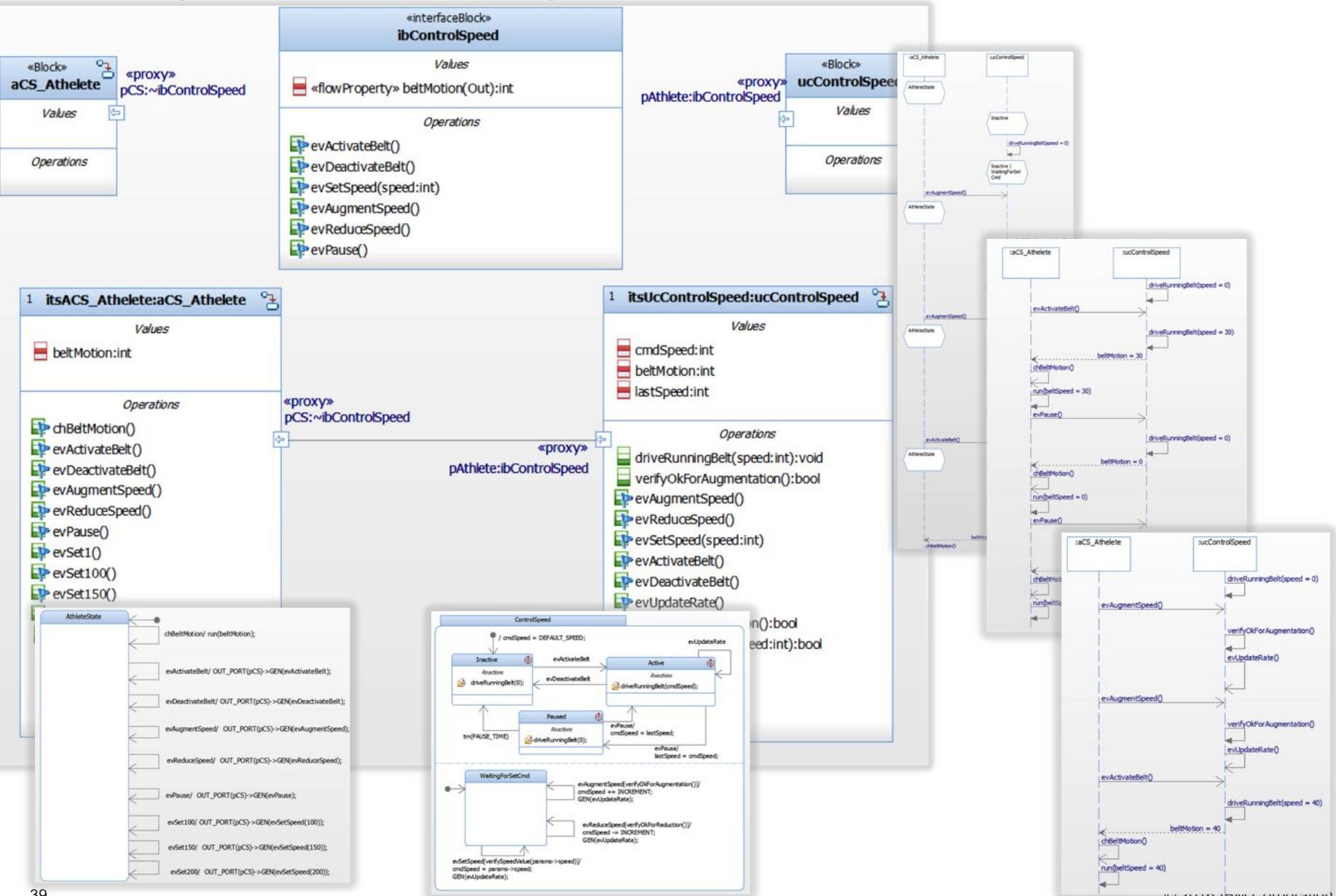
- Can the system be commanded into OFF directly?
- How are timing requirements verified?
- Reqs for degraded operation are now discovered to be inadequate – **add new reqs!**



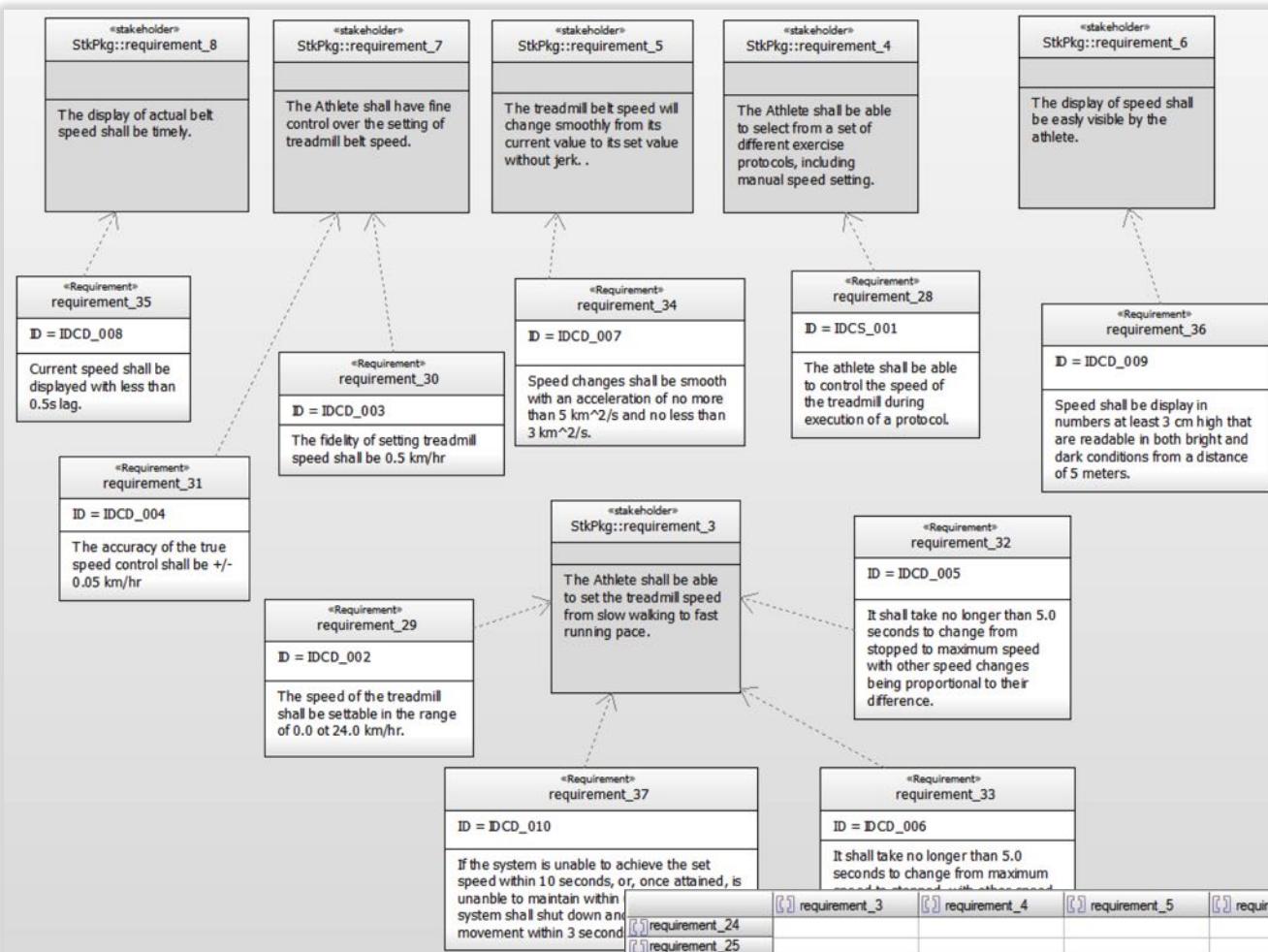
# Verify Requirements Adequacy Through Execution



# Identify gaps – add missing reqs – elaborate model - reexecute



# Add traceability

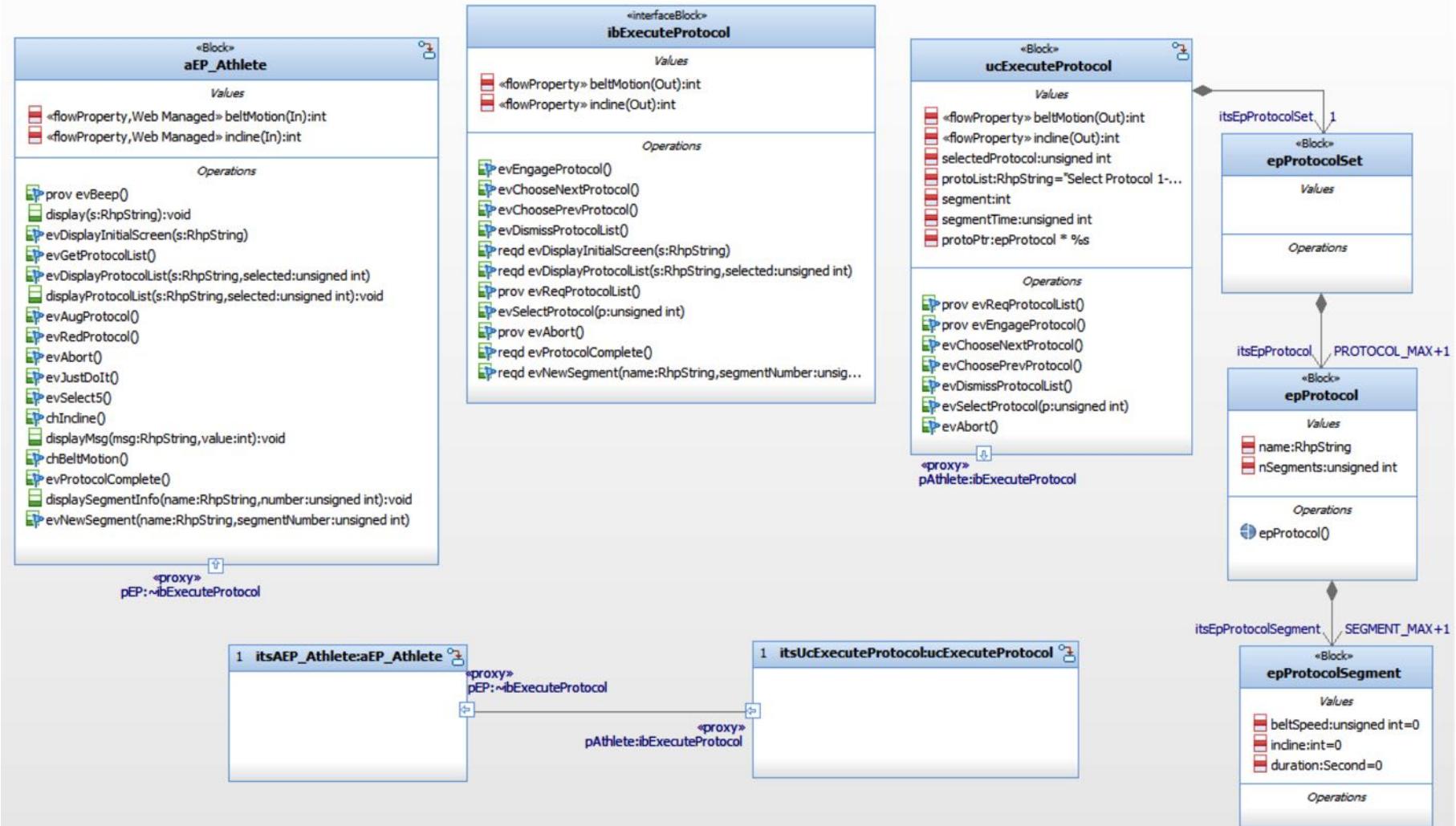


Stakeholder requirements

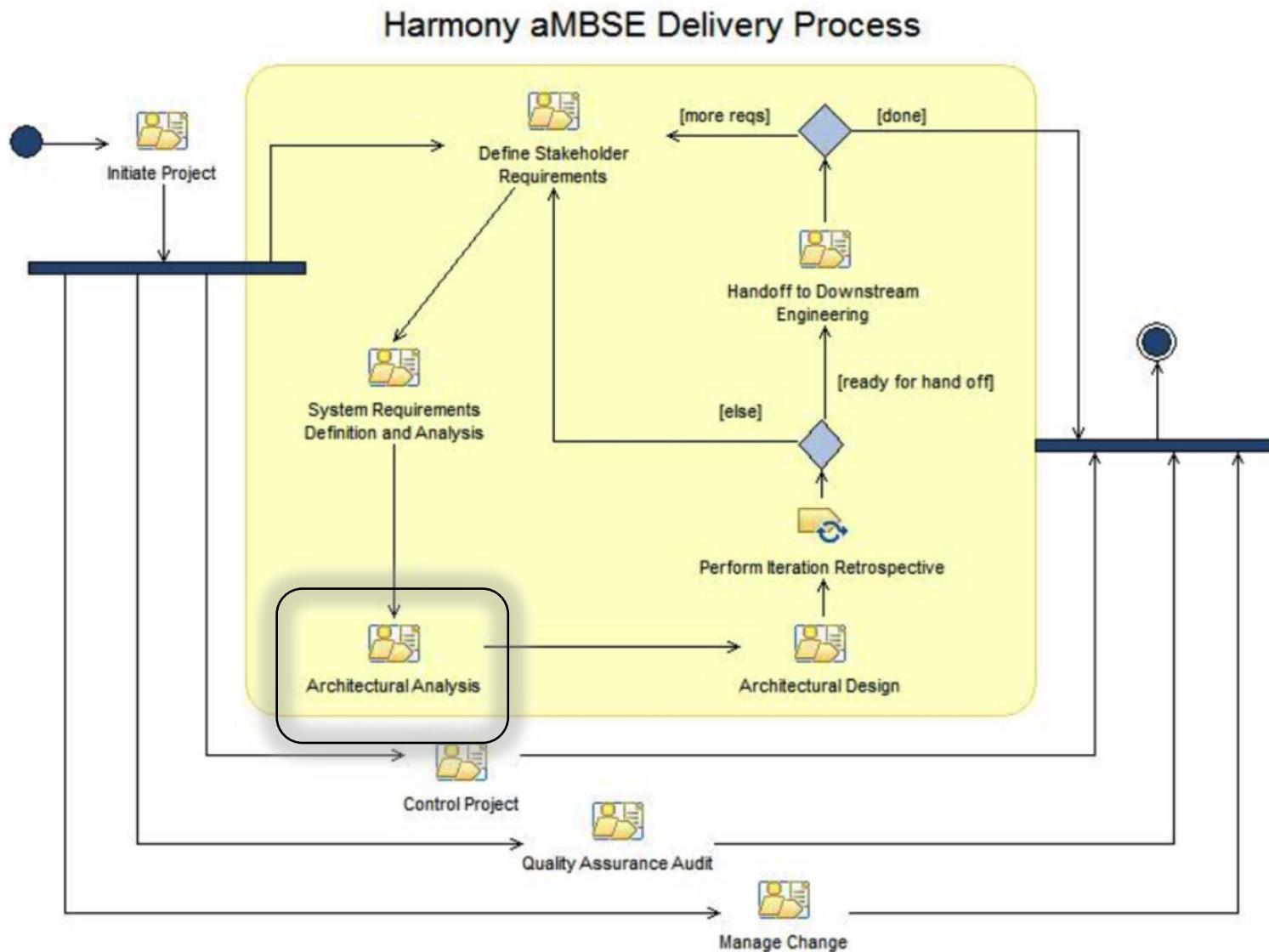
System requirements

Generated trace matrix

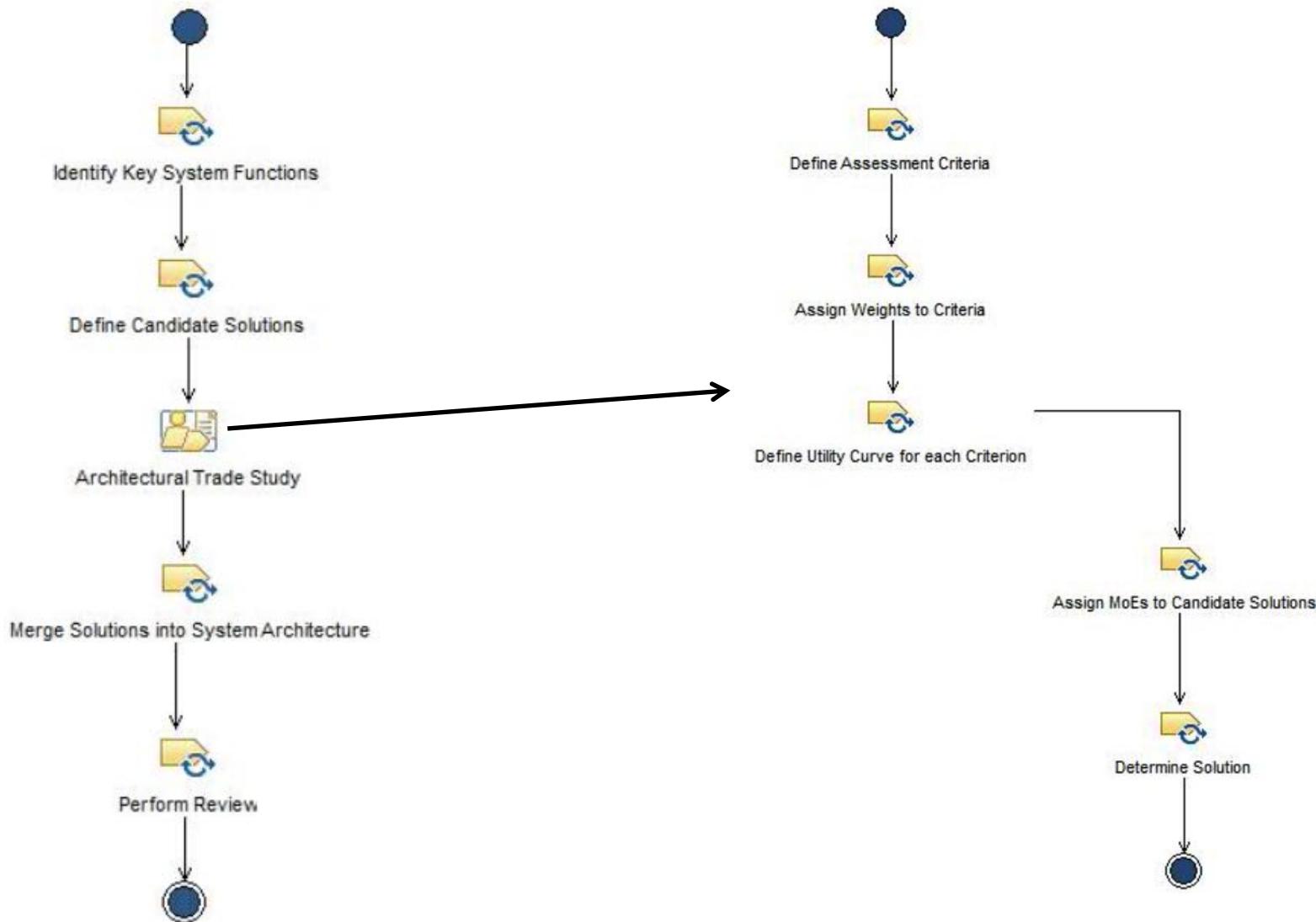
# Generate Logical Interfaces



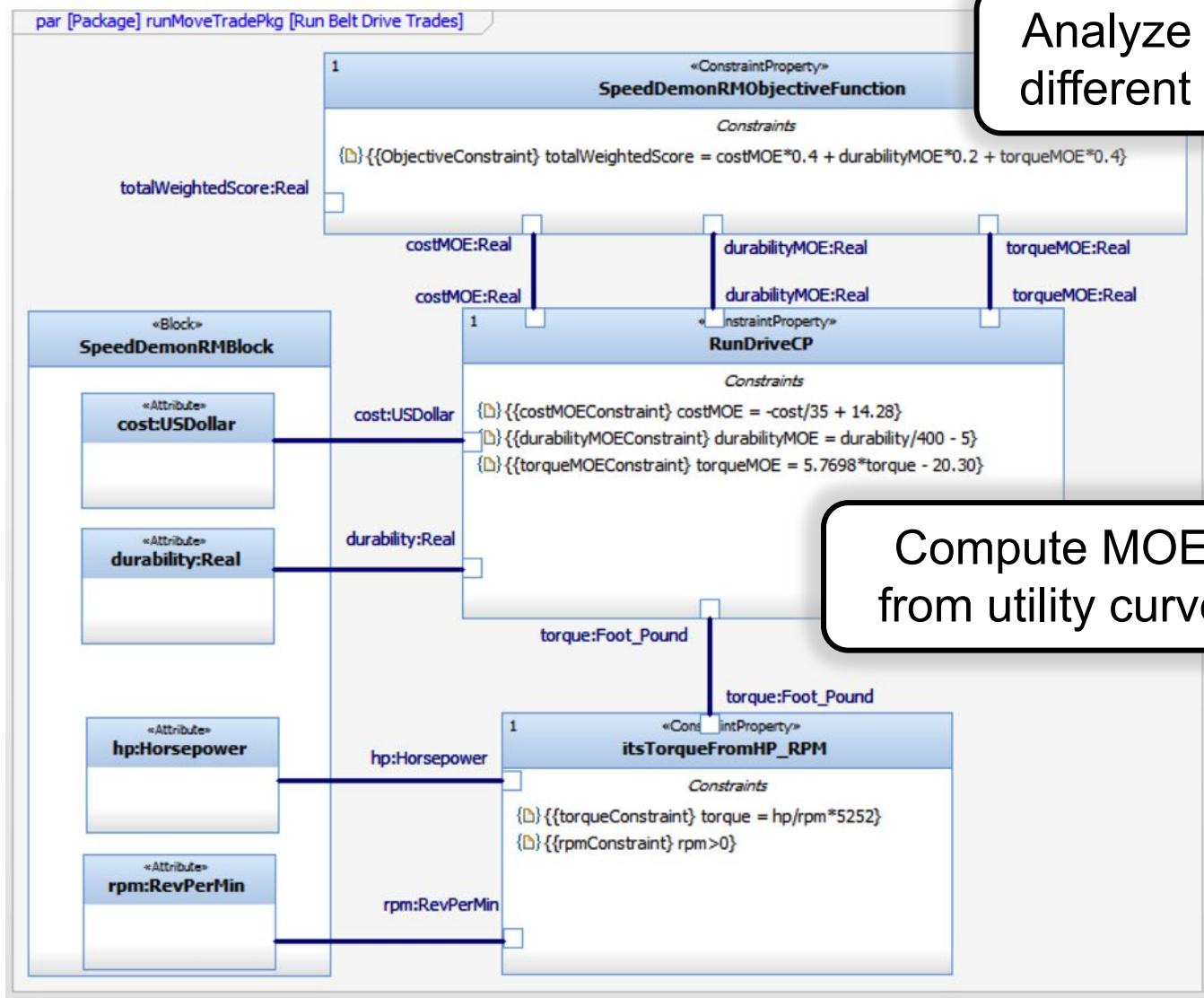
# Systems Architectural Analysis



# Systems Architectural Analysis



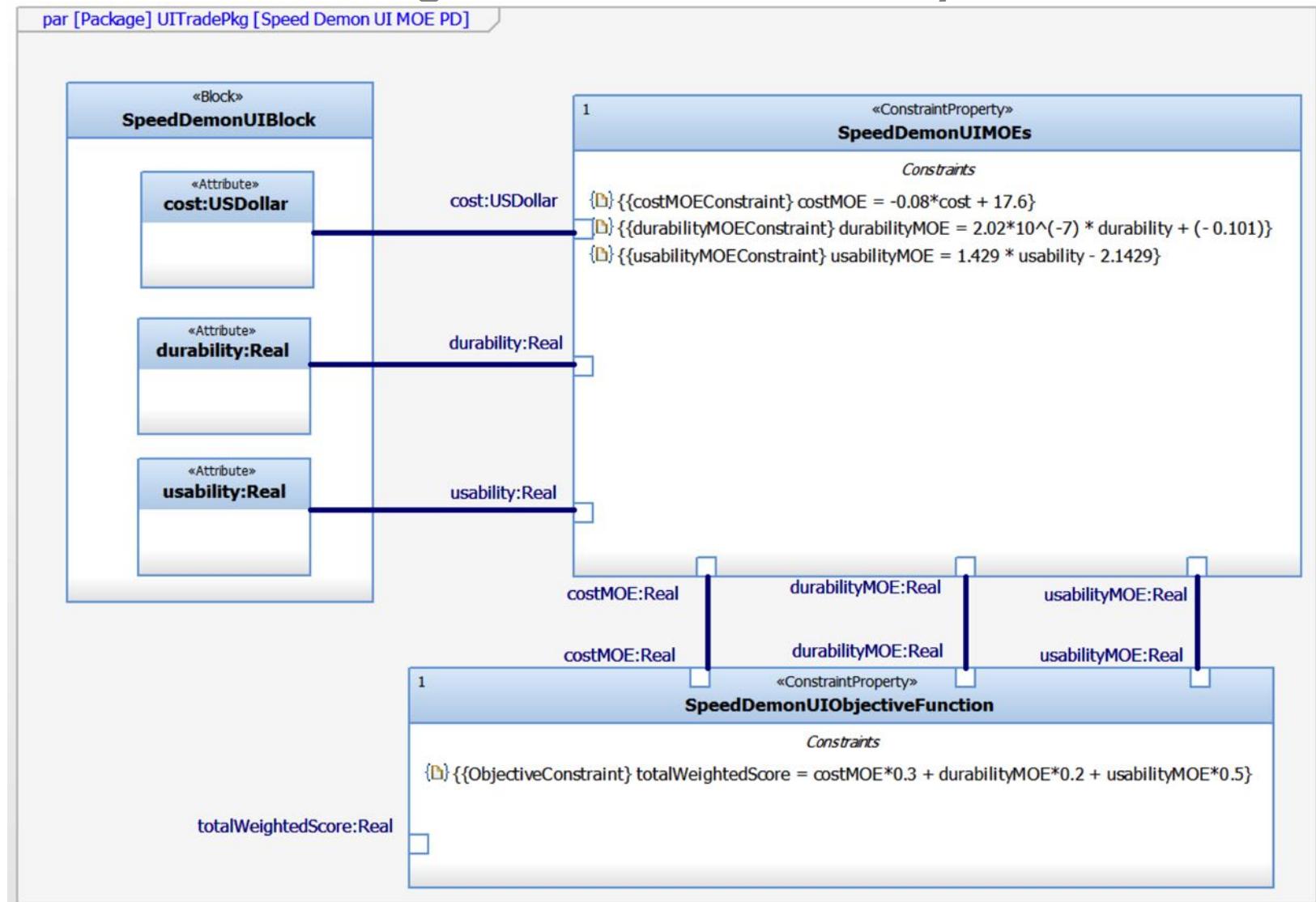
# SysML Parametric Diagram for Trades for SpeedDemon Motor



Analyze trades of different solutions

Compute MOEs from utility curves

# SysML Parametric Diagram for Trades for UI system



# Outputs of the trade analysis

The figure displays three software windows (PM\_1CV, PM\_2CV, and PM\_3CV) showing the results of a trade study for a pacemaker. Each window has a toolbar with Evaluate, Plot..., Refresh from Model, Update Model, Generate Report, Import Data..., Export Data..., and Export Constraints... buttons.

**PM\_1CV Data:**

Name	Type	Original Value	Value	Min.	Max.	Command
VOLUME_UPPER_LIMIT	Real	15.0	15.0			Fix
COST_UPPER_LIMIT	Real	250.00	250.00			Fix
MONTHS_UPPER_LIMIT	Real	120	120			Fix
Pacemaker						
cost	USDollar	150	150			Fix
deviceLifetime	Month	100	100			Fix
volume	CC	9.8	9.8			Fix
PacemakerMOEs						
cost	USDollar		150			
lifetime	Month		100			
volume	CC		9.8			
costMOE	Real		4			
lifetimeMOE	Real		8.333333333...			
volumeMOE	Real		3.466666666...			
COST_UPPER_LIMIT	Real		250.00			
MONTHS_UPPER_LIMIT	Real		120			
VOLUME_UPPER_LIMIT	Real		15.0			
costConstraint	Constraint	costMOE = 1...	costMOE = 1...			
lifeConstraint	Constraint	lifetimeMOE ...	lifetimeMOE ...			
volumeConstraint	Constraint	volumeMOE ...	volumeMOE ...			
PacemakerObjectiveFunc: PacemakerObjectiveFunc...						
costMOE	Real		4			
lifetimeMOE	Real		8.333333333...			
volumeMOE	Real		3.466666666...			
OverallScore	Real		5.626666666...			
ObjectiveFunction	Constraint	OverallScore ...	OverallScore ...			

Ready [4 free variable(s), 4 equation(s)]

**PM\_2CV Data:**

Name	Type	Original Value	Value	Min.	Max.	Command
VOLUME_UPPER_LIMIT	Real	15.0	15.0			Fix
COST_UPPER_LIMIT	Real	250.00	250.00			Fix
MONTHS_UPPER_LIMIT	Real	120	120			Fix
Pacemaker						
cost	USDollar	110	110			Fix
deviceLifetime	Month	80	80			Fix
volume	CC	6	6			Fix
PacemakerMOEs						
cost	USDollar		110			
lifetime	Month		80			
volume	CC		6			
costMOE	Real		5.600000000...			
lifetimeMOE	Real		6.666666666...			
volumeMOE	Real		6			
COST_UPPER_LIMIT	Real		250.00			
MONTHS_UPPER_LIMIT	Real		120			
VOLUME_UPPER_LIMIT	Real		15.0			
costConstraint	Constraint	costMOE = 1...	costMOE = 1...			
lifeConstraint	Constraint	lifetimeMOE ...	lifetimeMOE ...			
volumeConstraint	Constraint	volumeMOE ...	volumeMOE ...			
PacemakerObjectiveFunc: PacemakerObjectiveFunc...						
costMOE	Real		5.600000000...			
lifetimeMOE	Real		6.666666666...			
volumeMOE	Real		6			
OverallScore	Real		6.106666666...			
ObjectiveFunction	Constraint	OverallScore ...	OverallScore ...			

Ready [4 free variable(s), 4 equation(s)]

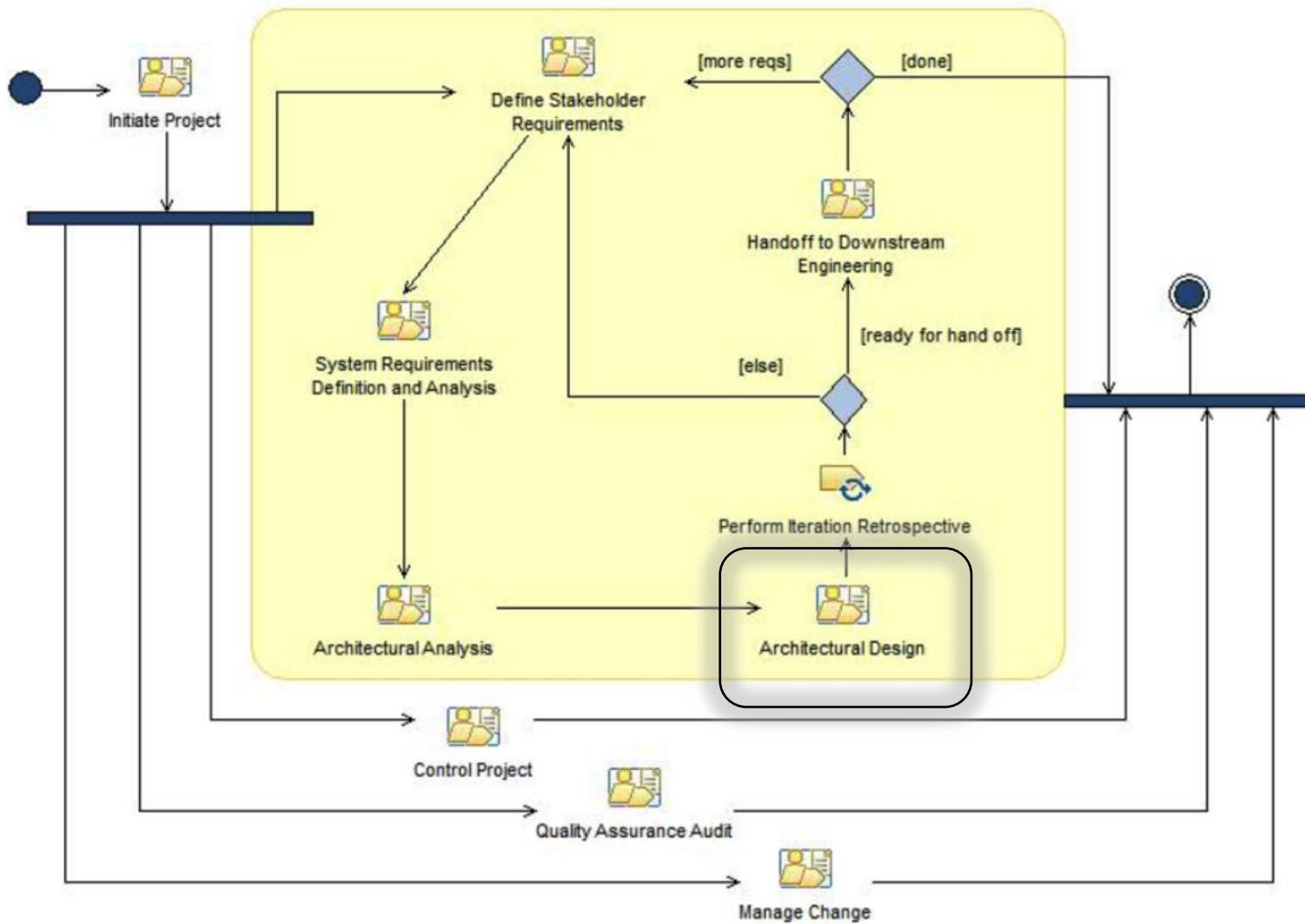
**PM\_3CV Data:**

Name	Type	Original Value	Value	Min.	Max.	Command
VOLUME_UPPER_LIMIT	Real	15.0	15.0			Fix
COST_UPPER_LIMIT	Real	250.00	250.00			Fix
MONTHS_UPPER_LIMIT	Real	120	120			Fix
Pacemaker						
cost	USDollar	250	250			Fix
deviceLifetime	Month	120	120			Fix
volume	CC	15	15			Fix
PacemakerMOEs						
cost	USDollar		250			
lifetime	Month		120			
volume	CC		15			
costMOE	Real		0			
lifetimeMOE	Real		10			
volumeMOE	Real		0			
COST_UPPER_LIMIT	Real		250.00			
MONTHS_UPPER_LIMIT	Real		120			
VOLUME_UPPER_LIMIT	Real		15.0			
costConstraint	Constraint	costMOE = 1...	costMOE = 1...			
lifeConstraint	Constraint	lifetimeMOE ...	lifetimeMOE ...			
volumeConstraint	Constraint	volumeMOE ...	volumeMOE ...			
PacemakerObjectiveFunc: PacemakerObjectiveFunc...						
costMOE	Real		0			
lifetimeMOE	Real		10			
volumeMOE	Real		0			
OverallScore	Real		4			
ObjectiveFunction	Constraint	OverallScore ...	OverallScore ...			

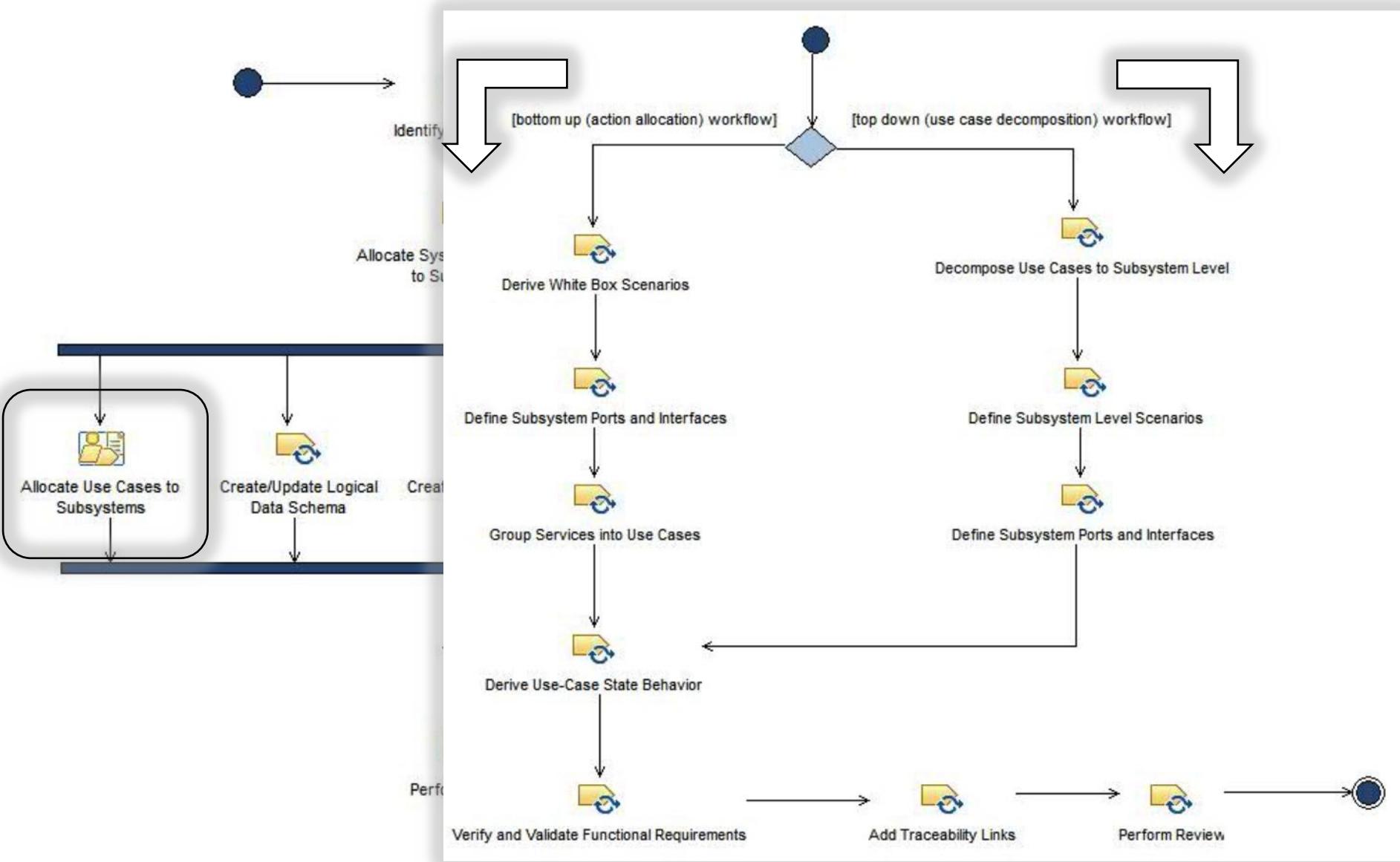
Ready [4 free variable(s), 4 equation(s)]

# Architectural Design

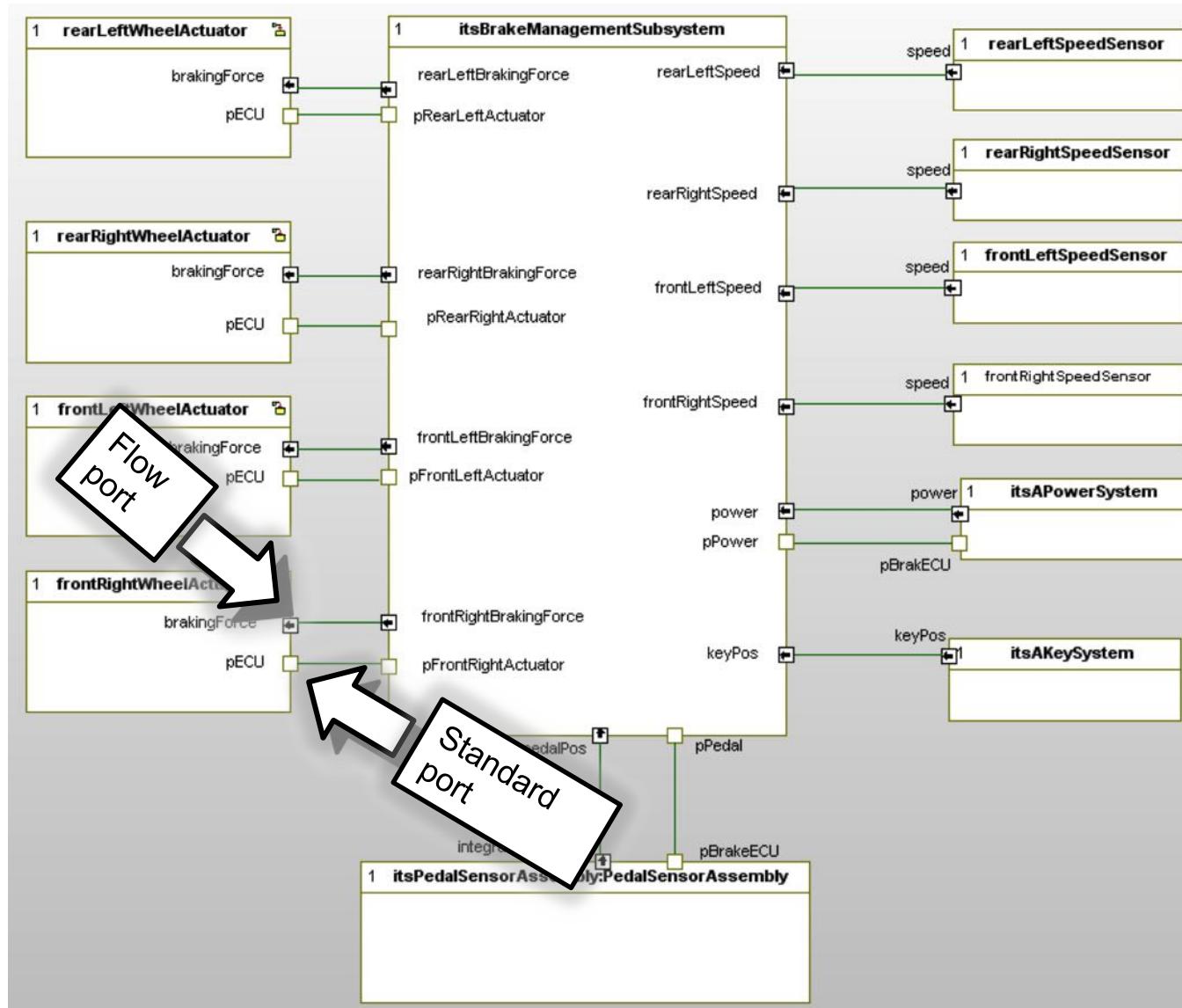
## Harmony aMBSE Delivery Process



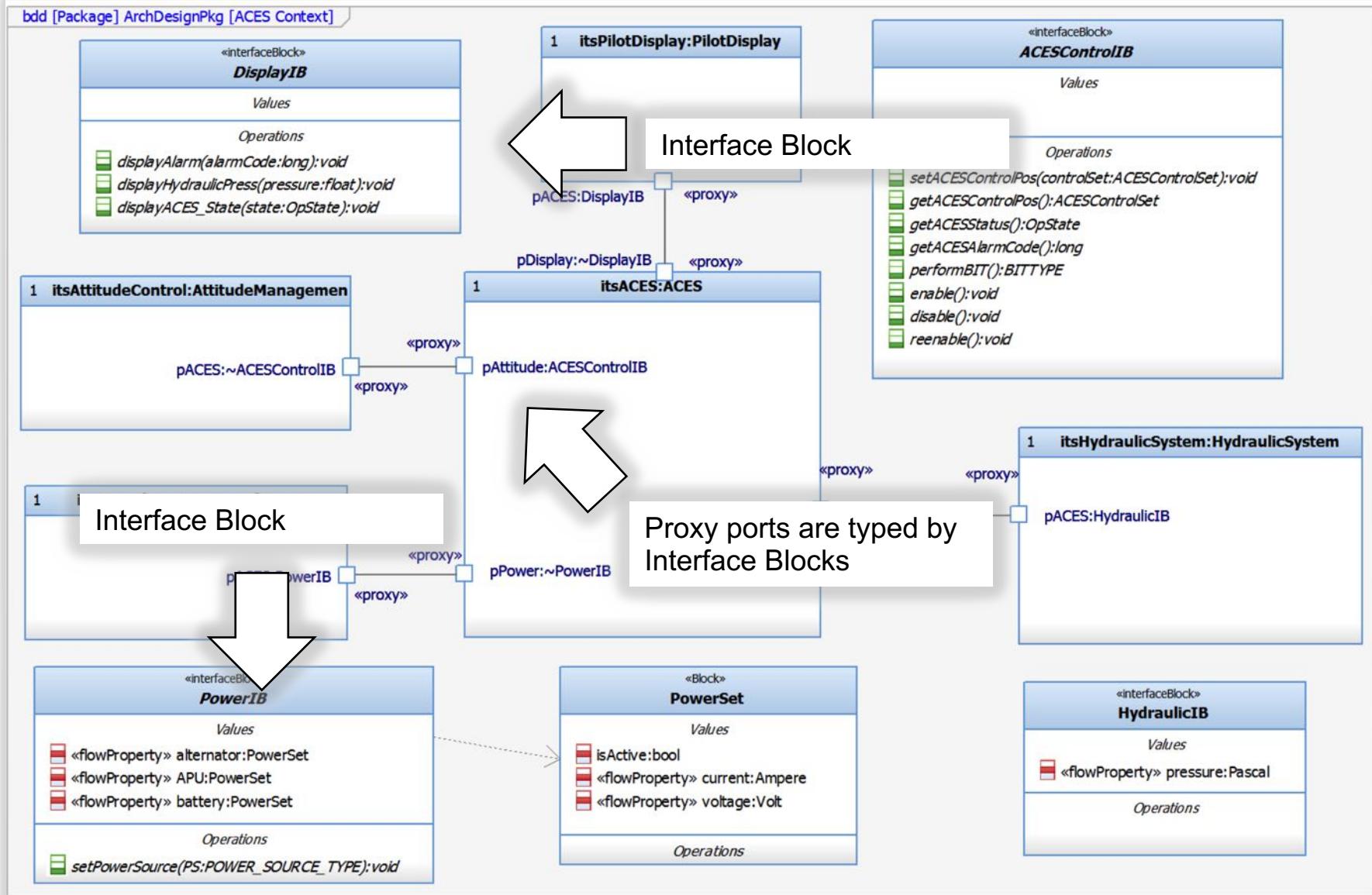
# Activity: Allocate Use Cases to Subsystems



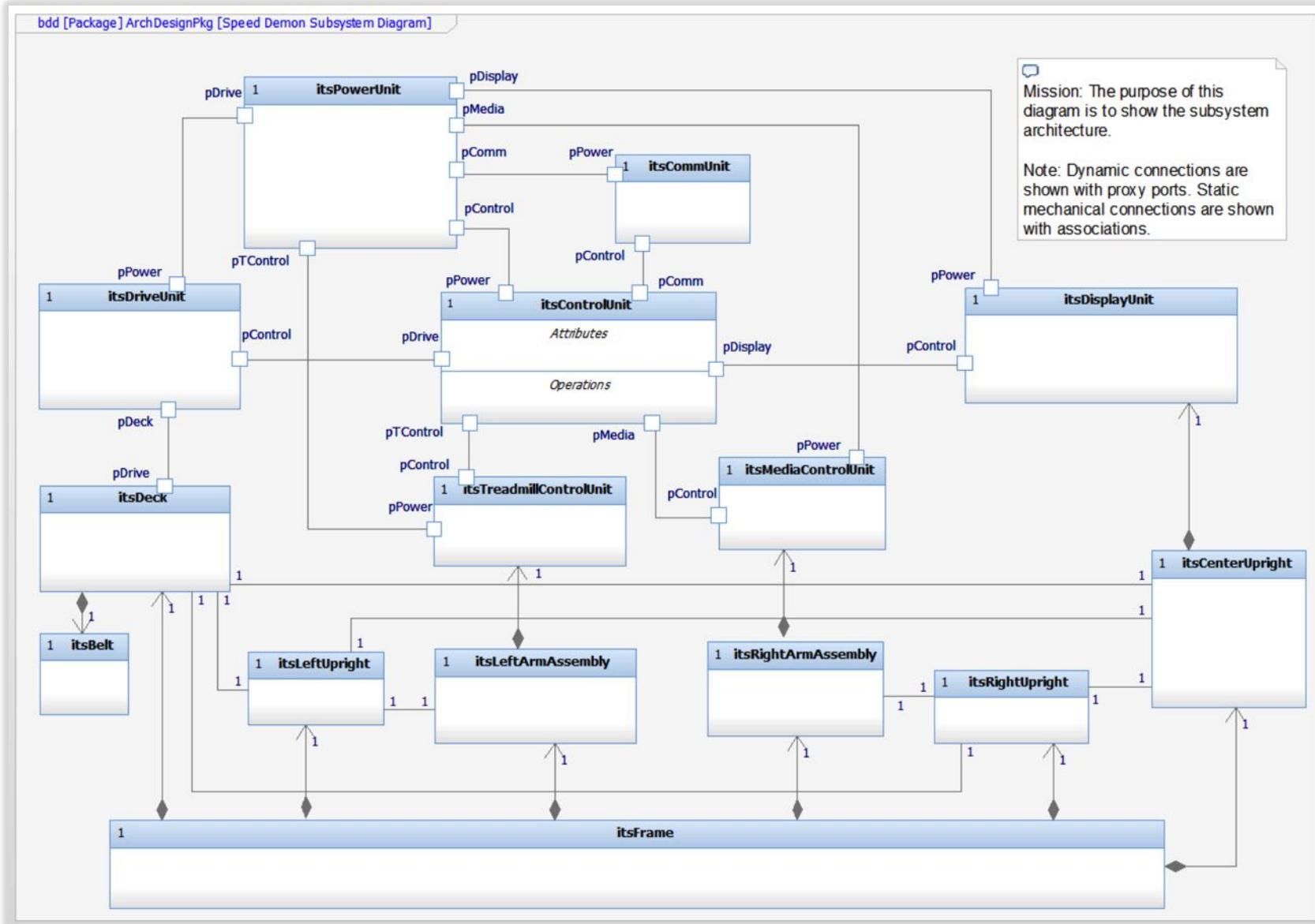
# Architecture View for Control Flow (UML & SysML 1.2)



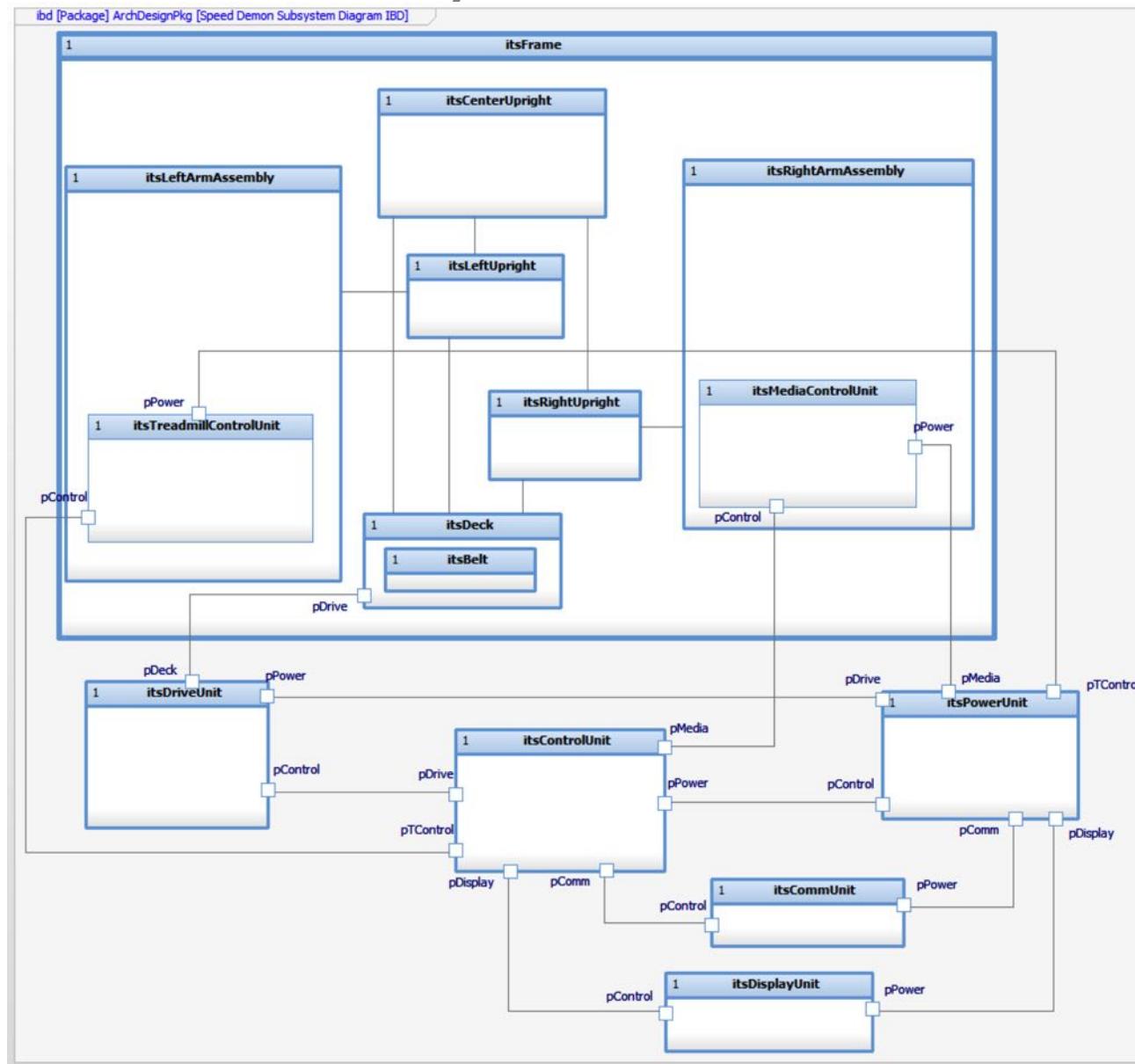
# Architecture View for Control Flow (SysML 1.3)



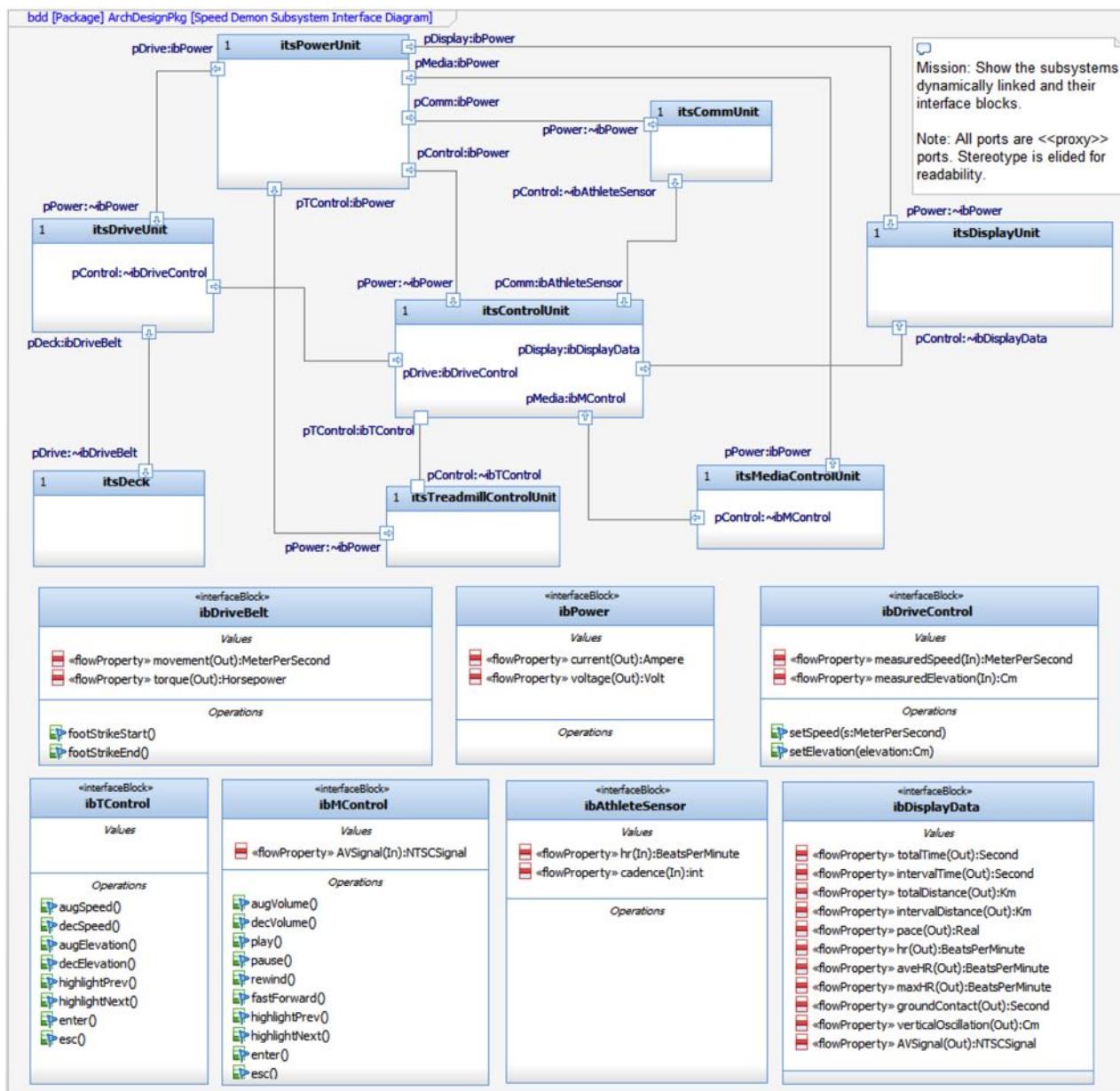
# Architectural View – Subsystem Diagram



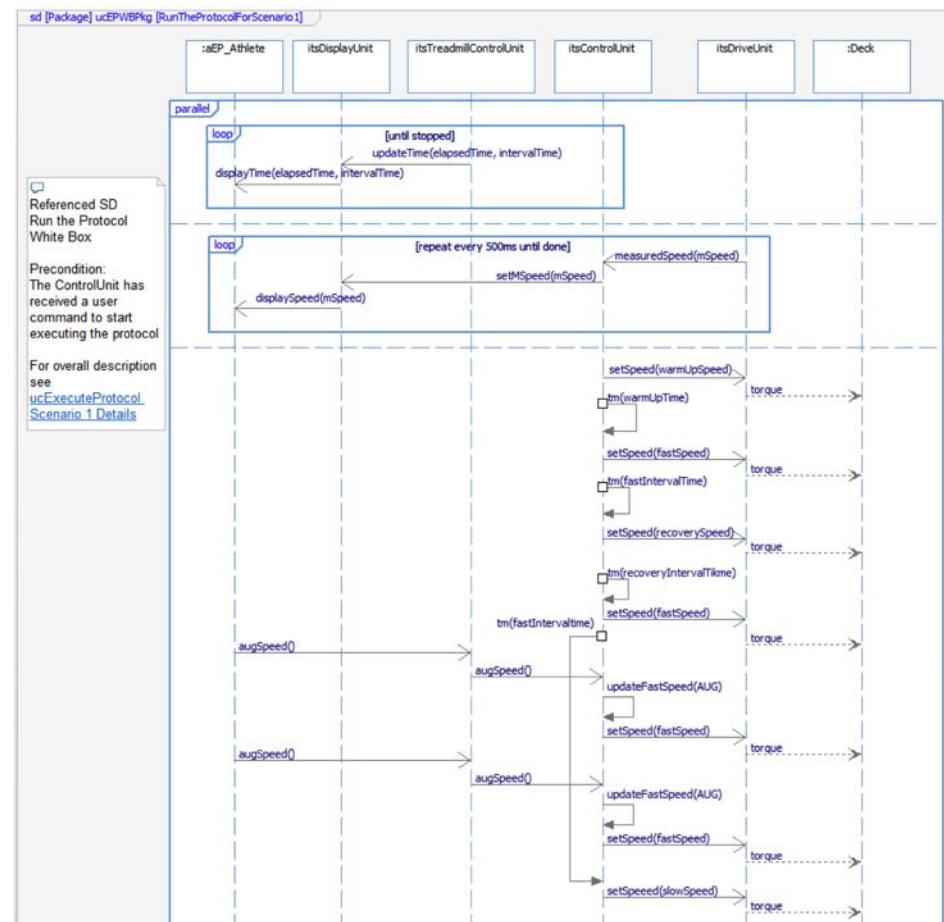
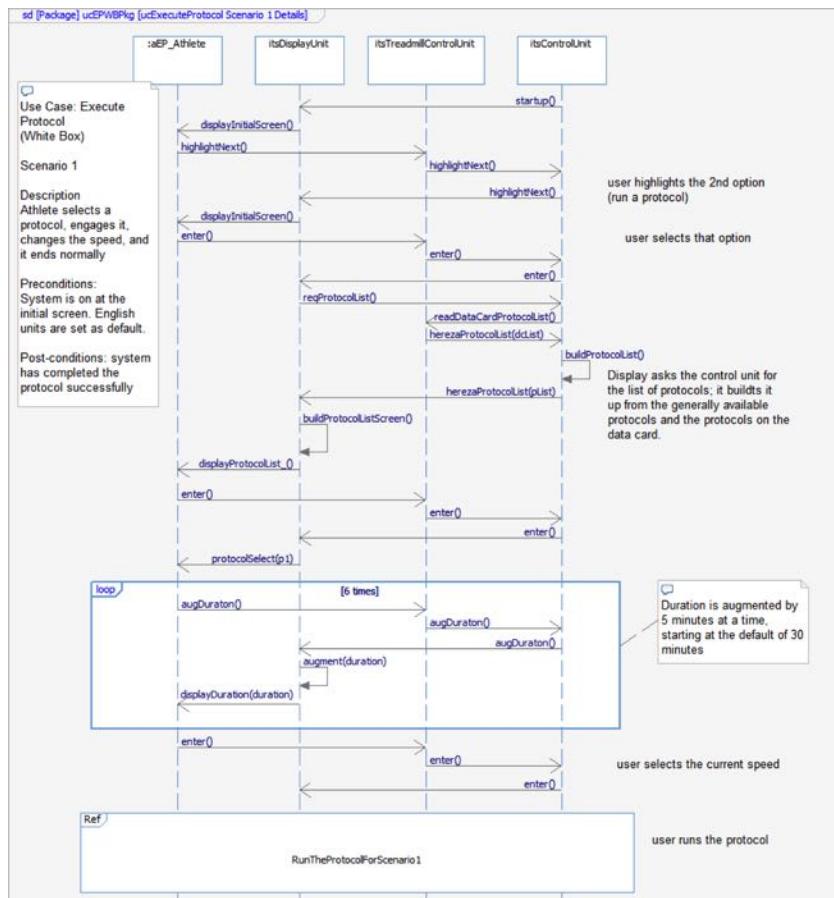
# Architectural View – Subsystem Detail Structure



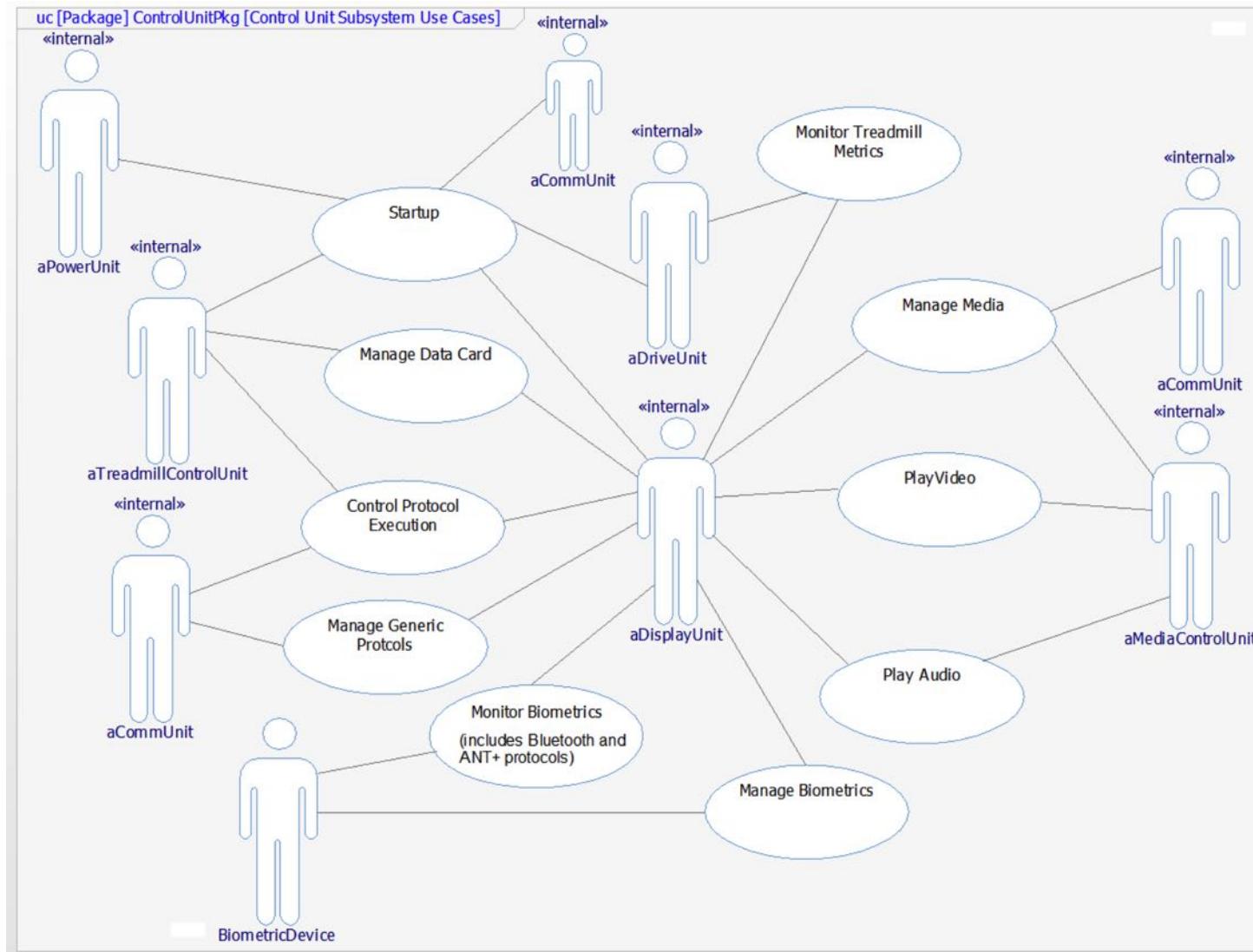
# Architecture with (logical) subsystem interfaces



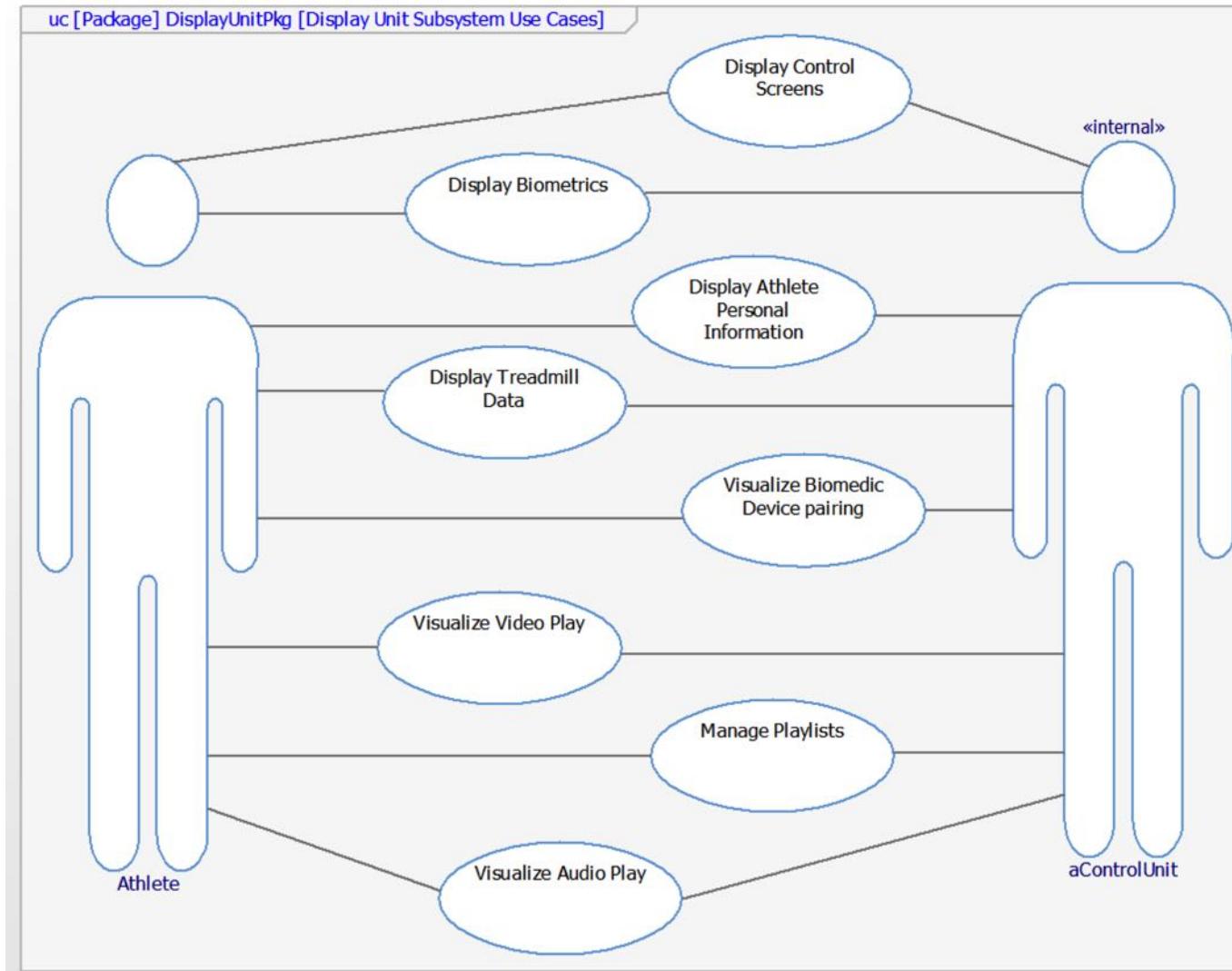
# Architectural Verification via Simulation/Execution



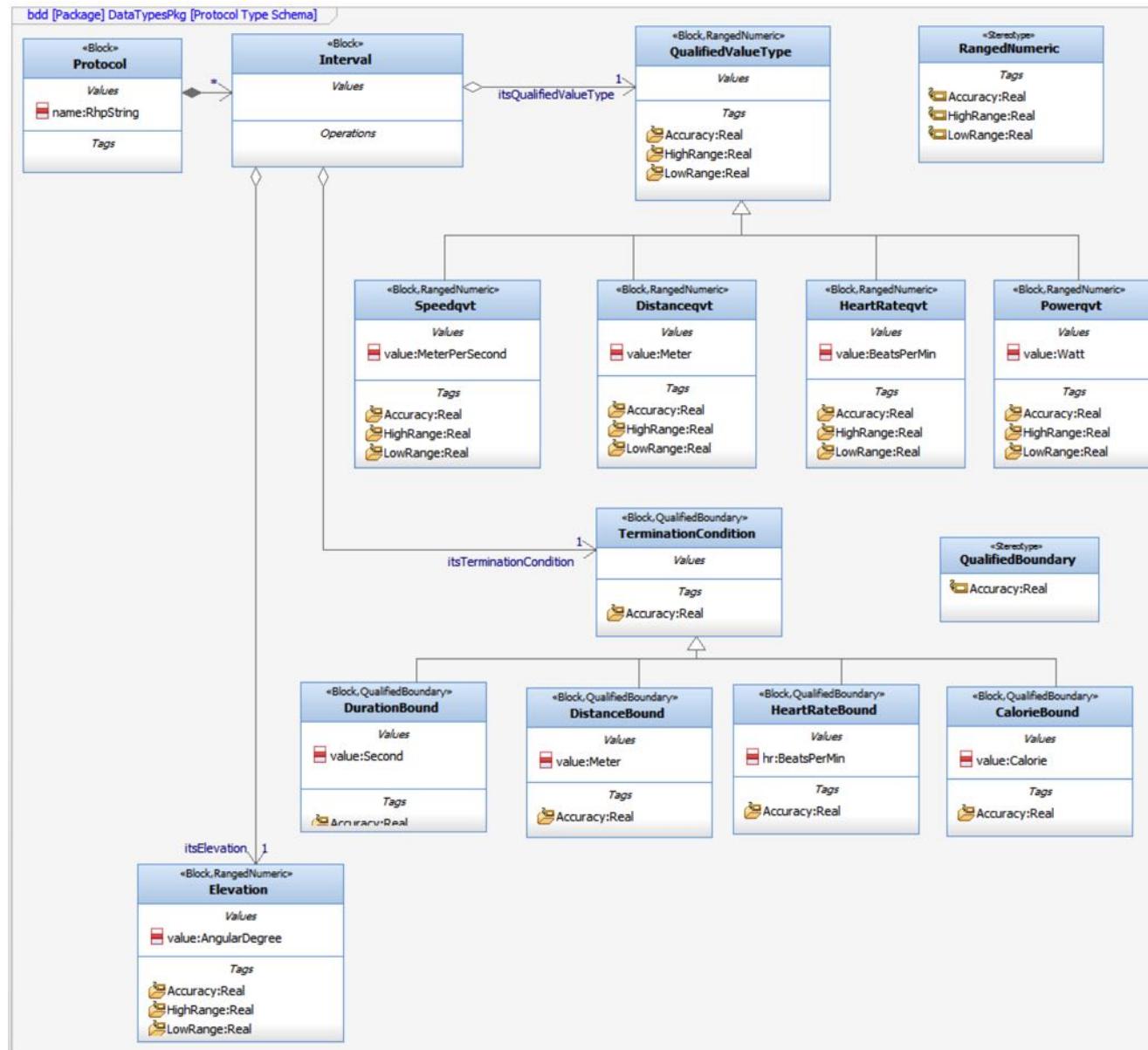
# Subsystem Specification (Control Unit)



# Subsystem Specification (Display Unit)



# Update / Elaborate Data/Flow Schema



# Capturing ICDs in the Model

- ICDs are not just a list of services but include:

- For each Service
  - Functional Description
  - Preconditions
  - Postconditions
  - Invariants
  - Performance
  - Error handling
  - Synchronization type
  - For each parameter

Description

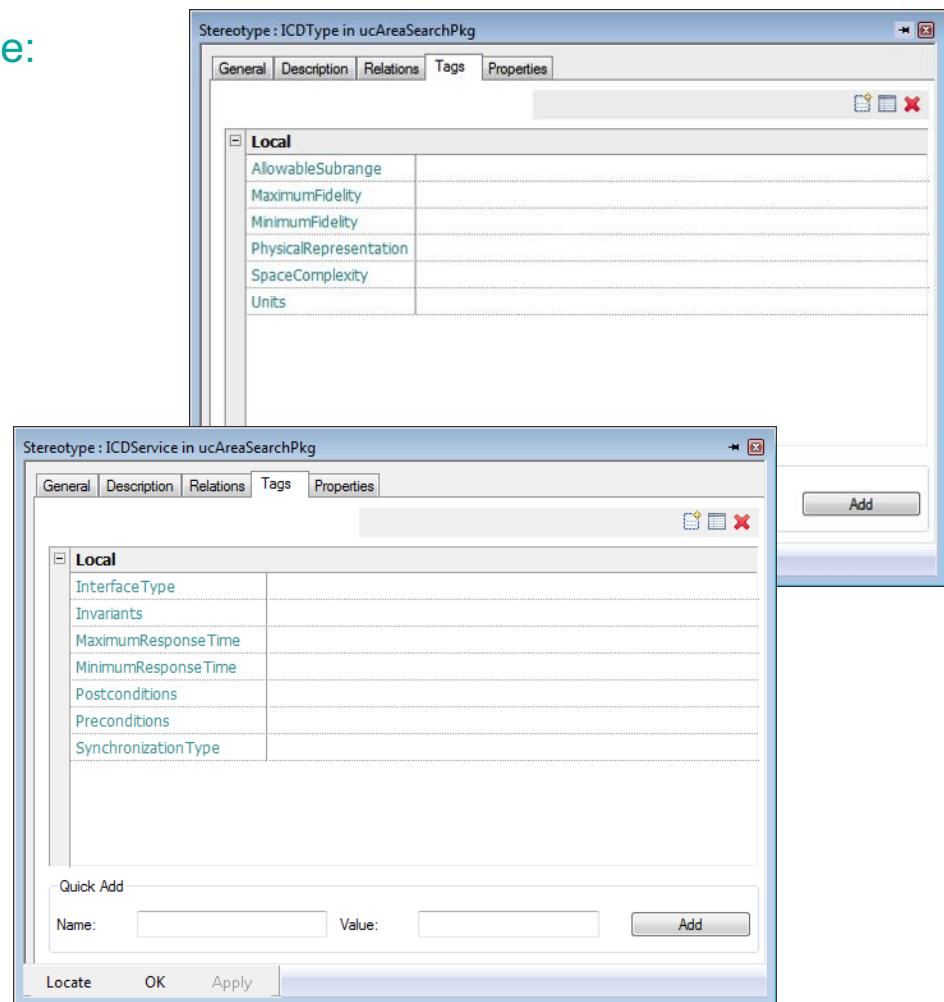
Type

Units

Valid subrange

Default value

- This metadata can be easily added as tags defined in stereotypes



# Capturing ICDs in the Model

The screenshot displays a modeling environment with several windows and diagrams:

- Diagram 1 (Top Left):** A system architecture diagram showing a central **pPAS** node connected to two **iPASPIlot** nodes and two **iPASFlight** nodes. One **iPASPIlot** is connected to a **pPilot** node, which in turn connects to an **itsU** node.
- Diagram 2 (Bottom Left):** A system architecture diagram showing a central **pPayloadOperator** node connected to an **iPASPayloadServices** node and an **iPASPayload** node.
- Mission Execution Window (Top Right):** A yellow-bordered window titled "Mission: Show execution" containing the text "«Interface,ICDInterface» iPASPilot".
- ICD Service List (Top Right):** A list of ICD services for the **iPASPilot** interface:
  - «ICDService» remotePilotingSystemTests(status:StatusType)
  - «ICDService» preflightTests(status:StatusType)
  - «ICDService» evControlSurfaces(rudderPos:int,elevatorPos:int,lAileronPos:int,rAileronPos:int,lWingFlapPos:int,rWingFlapPos:int)
  - «ICDService» evAttitude(roll:int,pitch:int,yaw:int)
  - «ICDService» evPosition(longitude:int,latitude:int,altitude:int)
  - «ICDService» evEngineSpeed(speed:SpeedType)
  - «ICDService» evGroundSpeed(speed:SpeedType)
  - «ICDService» evAirSpeed(speed:SpeedType)
- Properties Window (Bottom Left):** A "Type : SpeedType in ucAreaSearchPkg" properties window with tabs for General, Description, Declaration, Relations, Tags, and Properties. The Properties tab shows the following data for the **ucAreaSearchPkg** ICD Type:
 

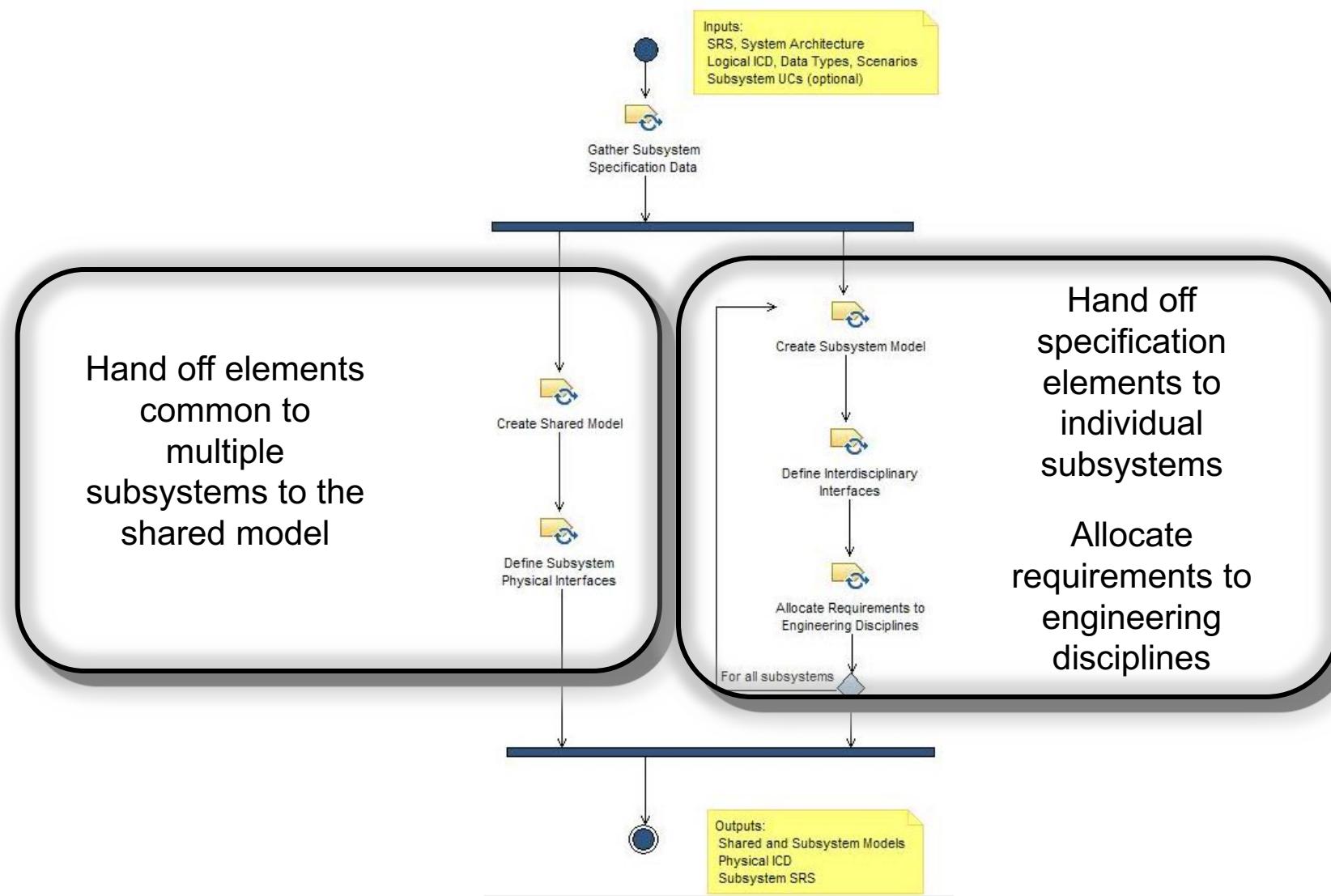
AllowableSubrange	0 .. 2000
MaximumFidelity	1 mph
MinimumFidelity	0.1 mph
PhysicalRepresentation	32 bit signed integer
SpaceComplexity	32 bits
Units	mph
- Properties Window (Bottom Right):** A "Reception : remotePilotingSystemTests in iPASPilot" properties window with tabs for General, Description, Relations, Tags, and Properties. The Properties tab shows the following data for the **remotePilotingSystemTests** ICD Service:
 

InterfaceType	Asynchronous event
Invariants	
MaximumResponseTime	10s
MinimumResponseTime	
Postconditions	results of system tests returned
Preconditions	system is operational
SynchronizationType	asynchronous

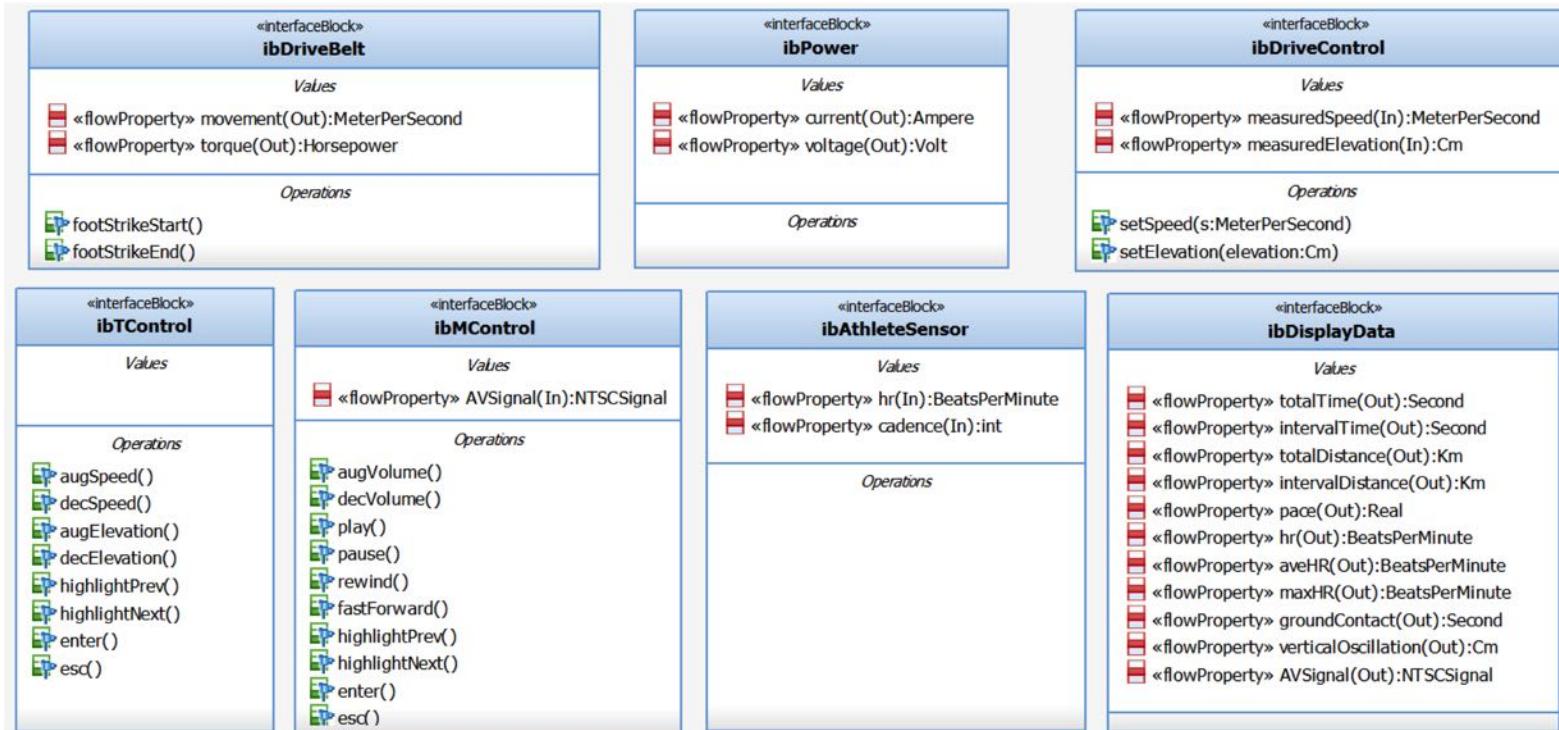
**Text Overlay:**

Use flow ports for communication  
Use standard ports for control

# Hand off Workflow

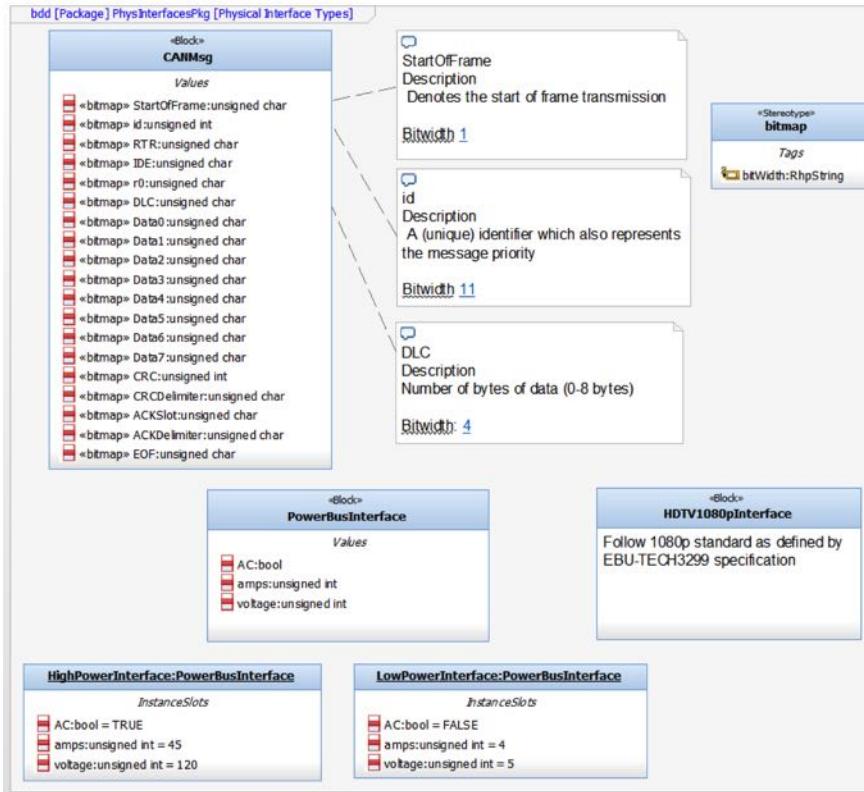


# Convert Logical to Physical Interfaces



Logical Interfaces

# SpeedDemon Physical Interfaces (using CAN™ Bus)

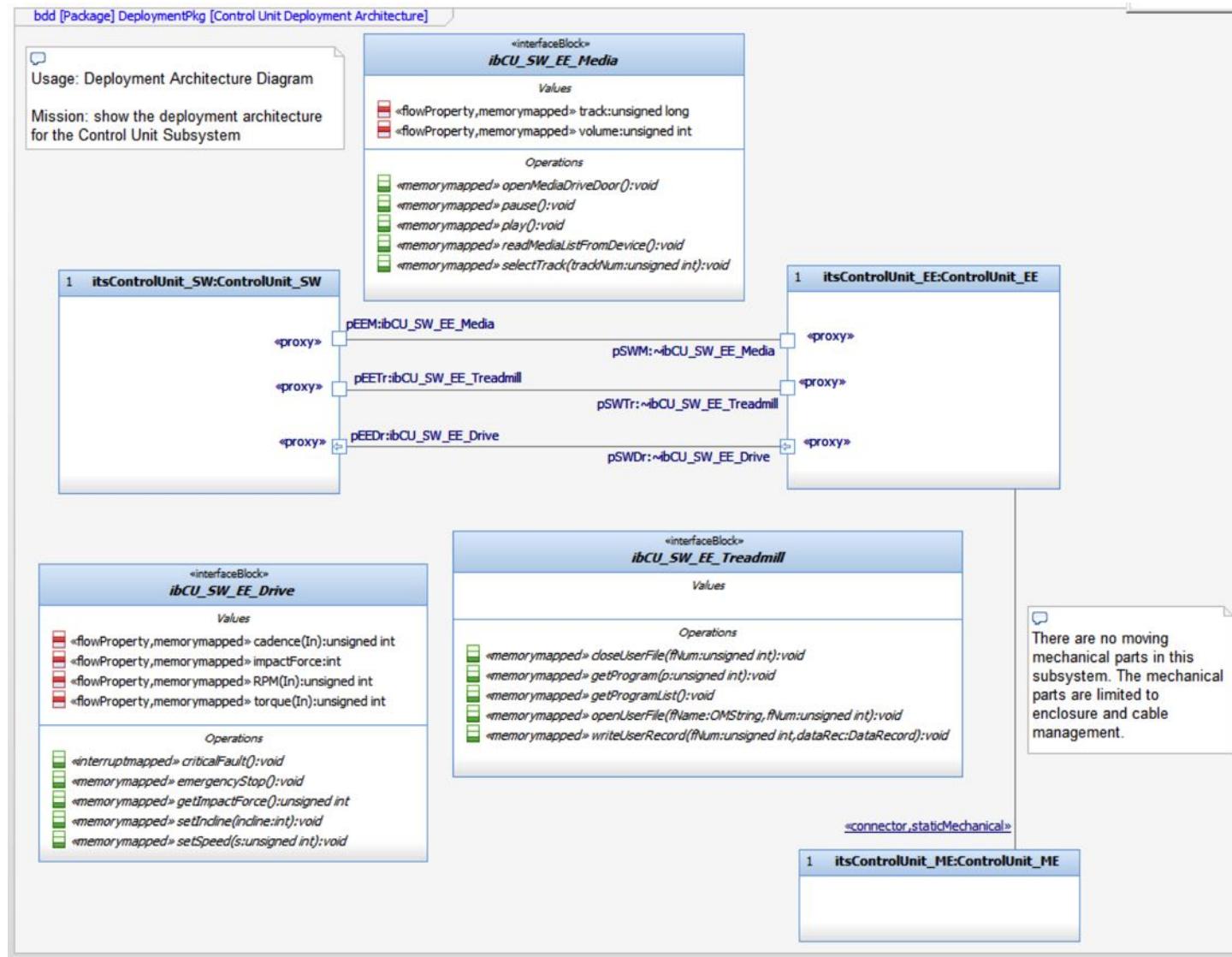


## Interface structure specification



## Instance specifications for messages

# Subsystem Deployment Architecture (SW-EE-ME)



# Instance Specs for SW-EE Interfaces in Control Unit

**bdd [Package] DeploymentPkg [Details of ibCU\_SW\_EE\_Drive interface block]**

Mission: show the interface details for the SW-EE interface for drive control in the Control Unit Subsystem

**ibCU\_SW\_EE\_Drive**

**Values**

- «flowProperty, memorymapped» cadence(*In*):unsigned int
- «flowProperty, memorymapped» impactForce:int
- «flowProperty, memorymapped» RPM(*In*):unsigned int

**Operations**

- «interruptmapped» criticalFault():void
- «memorymapped» emergencyStop():void
- «memorymapped» getImpactForce():unsigned int
- «memorymapped» setIncline(incline:int):void
- «memorymapped» setSpeed(s:unsigned int):void

**impactForce:int**

Returns impact force in Kg

**Tags**

- Bitmap:RhpString=A000:0014-A000:0017 4 byte unsigned int
- direction:FlowDirection=Unspecified
- Numer\_Of\_Bytes:RhpString=4
- Range\_High:RhpReal=500
- Range\_Low:RhpReal=0
- Start\_Address:RhpString=A000:0014
- Timing\_Constraints:RhpString
- Usage:RhpString=Read only access

**RPM(*In*):unsigned int**

Returns the current measured drive RPM

**Tags**

- Bitmap:RhpString=A000:0010-A000:0013 4 byte unsigned int
- direction:FlowDirection=In
- Numer\_Of\_Bytes:RhpString=4
- Range\_High:RhpReal=10,000
- Range\_Low:RhpReal=0
- Start\_Address:RhpString=A000:0010
- Timing\_Constraints:RhpString
- Usage:RhpString=Read only access

**setSpeed(s:unsigned int):void**

This command to the electronics sets the speed of the drive unit controlling the belt.

**Tags**

- Bitmap:RhpString=A000:0000 - A000:0003 Write 4 byte speed...
- Numer\_Of\_Bytes:RhpString=4
- Range\_High:RhpReal=0
- Range\_Low:RhpReal=800
- Start\_Address:RhpString=A000:0000
- Timing\_Constraints:RhpString
- Usage:RhpString=Write the speed value in the 4 byte block. ...

**Bitmap: A000:0000 - A000:0003 Write 4 byte speed value**

**Usage:** Write the speed value in the 4 byte block. Reading the value reads the current speed of the treadmill (note: this is not necessarily the value just written)  
Speed is in units (0.01 m/sec) so a value of 671 is 6.71 m/sec

**setIncline(incline:int):void**

The command to the electronics sets the degree of incline of the deck. Value is 10° the angle in degrees (e.g. value of -35 means an angle of -3.5 degrees)

**Tags**

- Bitmap:RhpString=A000:0006-A000:0007 2 byte integer ...
- Numer\_Of\_Bytes:RhpString=2
- Range\_High:RhpReal=200
- Range\_Low:RhpReal=-100
- Start\_Address:RhpString=A000\_0006
- Timing\_Constraints:RhpString
- Usage:RhpString=Write sets the commanded value; read ...

**emergencyStop():void**

Writing a nonzero value causes an emergency stop. Writing #0000 re-enables movement.

**Tags**

- Bitmap:RhpString=A000:0008-A000:0009 2 byte unsigned int
- Numer\_Of\_Bytes:RhpString=2
- Range\_High:RhpReal=0
- Range\_Low:RhpReal=FFFF
- Start\_Address:RhpString=A000:0008
- Timing\_Constraints:RhpString=Emergency stop en...
- Usage:RhpString=Non-zero write engages functio...

**cadence(*In*):unsigned int**

Returns the currently measured cadence value in 2 bytes with units of steps/min (of both legs)

**Tags**

- Bitmap:RhpString=A000:000A-A000:000B 2 byte unsigned int
- direction:FlowDirection=In
- Numer\_Of\_Bytes:RhpString=2
- Range\_High:RhpReal=220
- Range\_Low:RhpReal=0
- Start\_Address:RhpString=A000:000C
- Timing\_Constraints:RhpString
- Usage:RhpString=Read only access

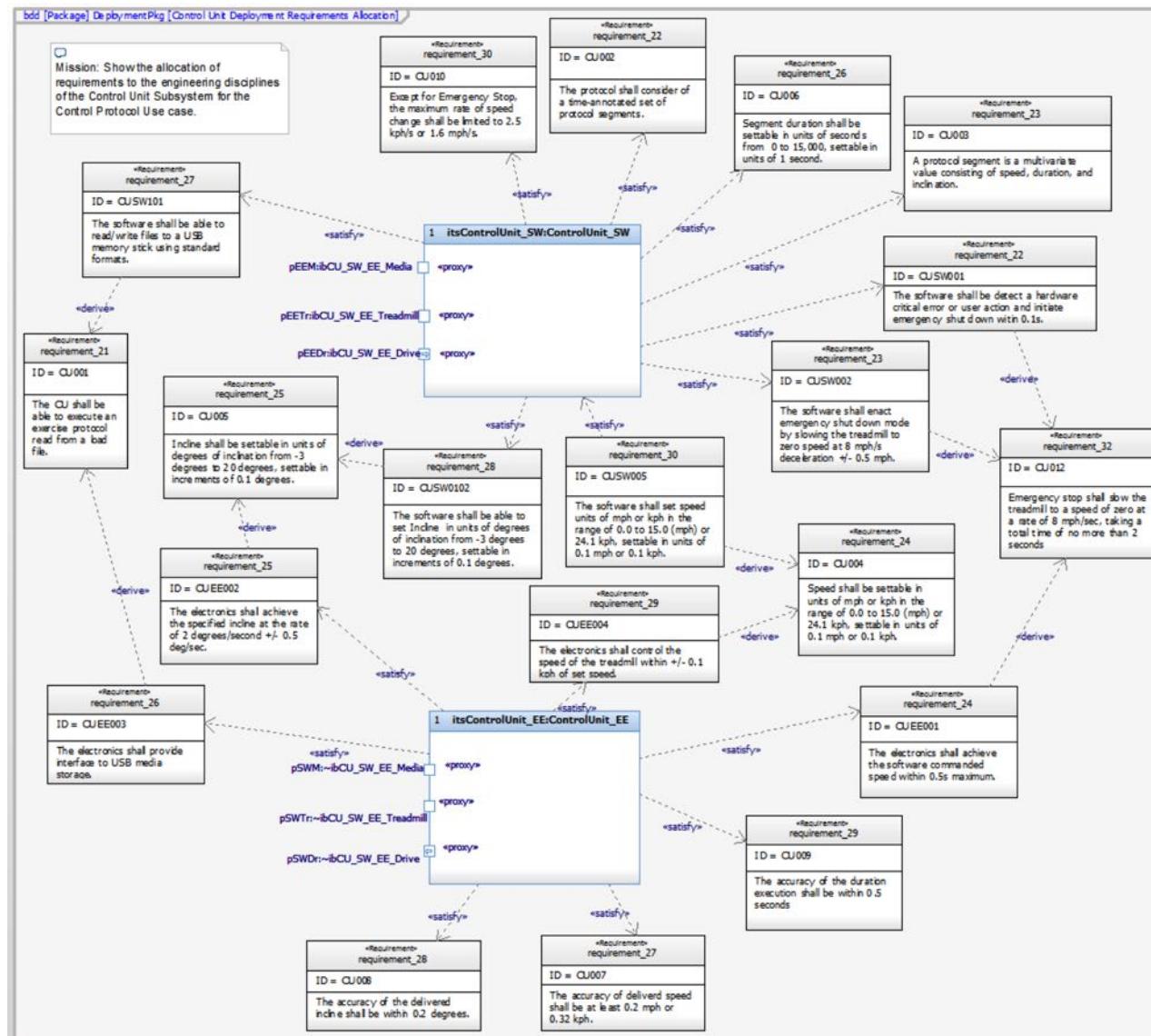
**criticalFault():void**

This electronic service generates an interrupt when a critical fault occurs. The data at the specified data address is the 2-byte error code.

**Tags**

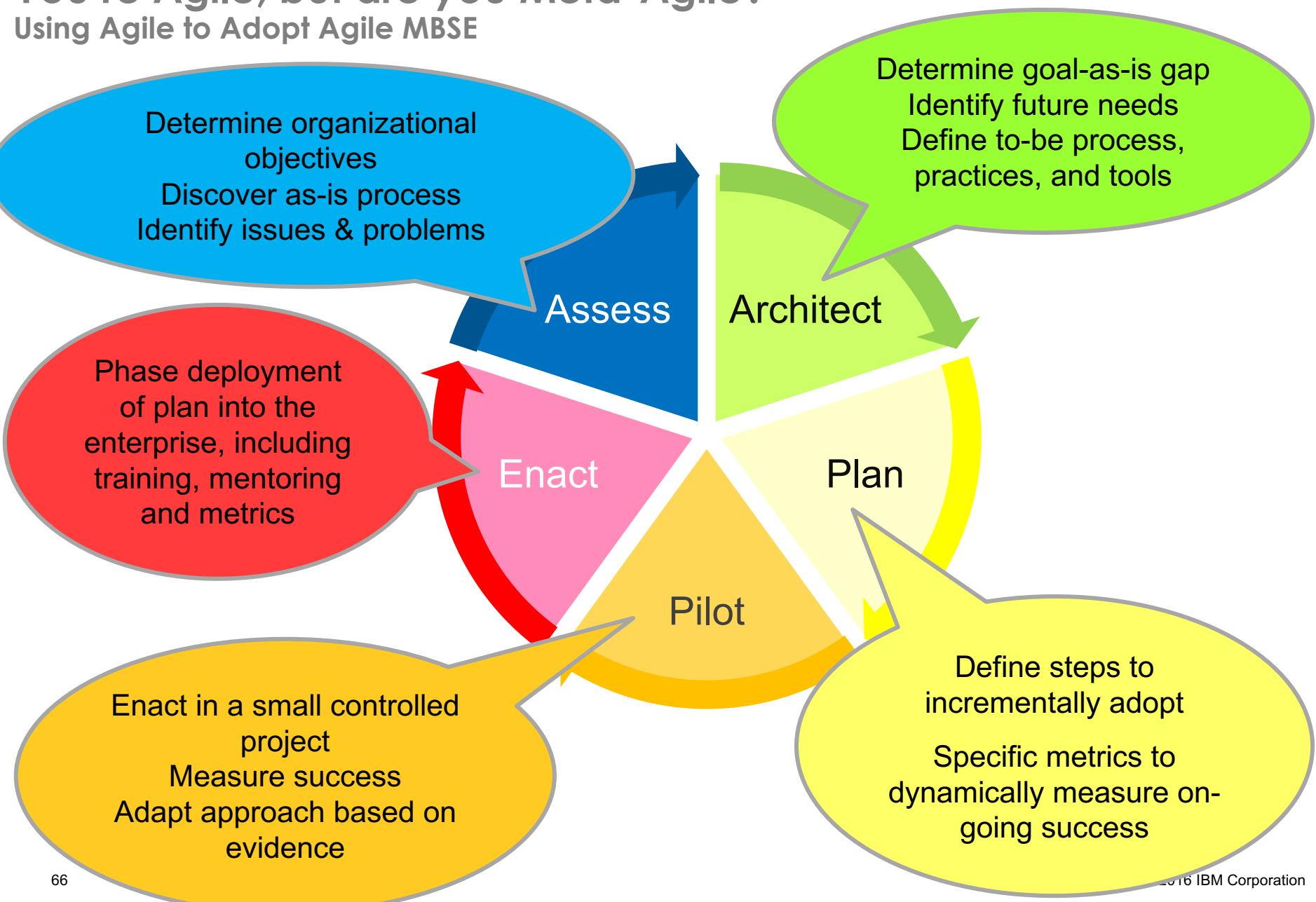
- Byte\_Width:RhpString=2
- Data\_Address:RhpString=A100:0000
- Data\_Field\_Type:RhpString=unsigned integer
- Interrupt\_number:RhpString=0x018

# Allocation on Subsystem requirements to engineering disciplines



# You're Agile, but are you Meta-Agile?

## Using Agile to Adopt Agile MBSE



# Summary

MBSE provides precision and verifiability to the SE Process

Agile methods add quality, responsiveness and adaptability to the process

Continuous verification allows you to avoid costly defects

Adopt Agile in an incremental, measured fashion for best results

IBM's Rational tooling supports MBSE and agile methods

IBM's Harmony Process defines an agile MBSE process with industry best practices

# References

