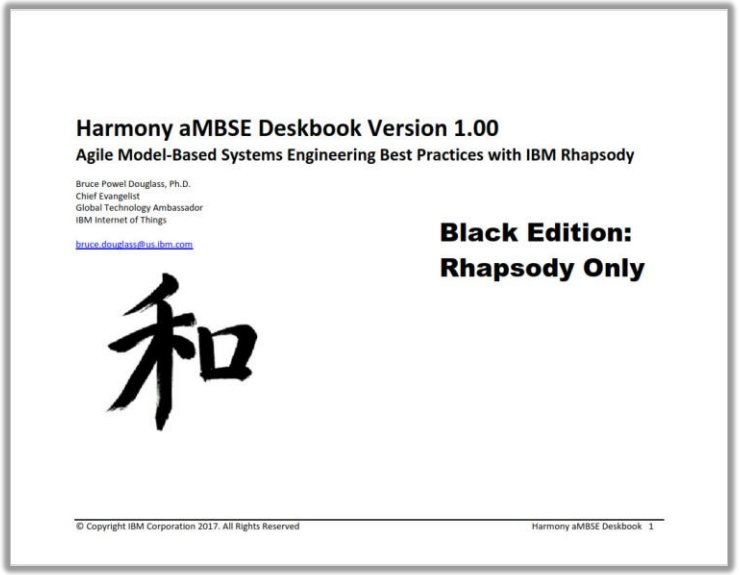
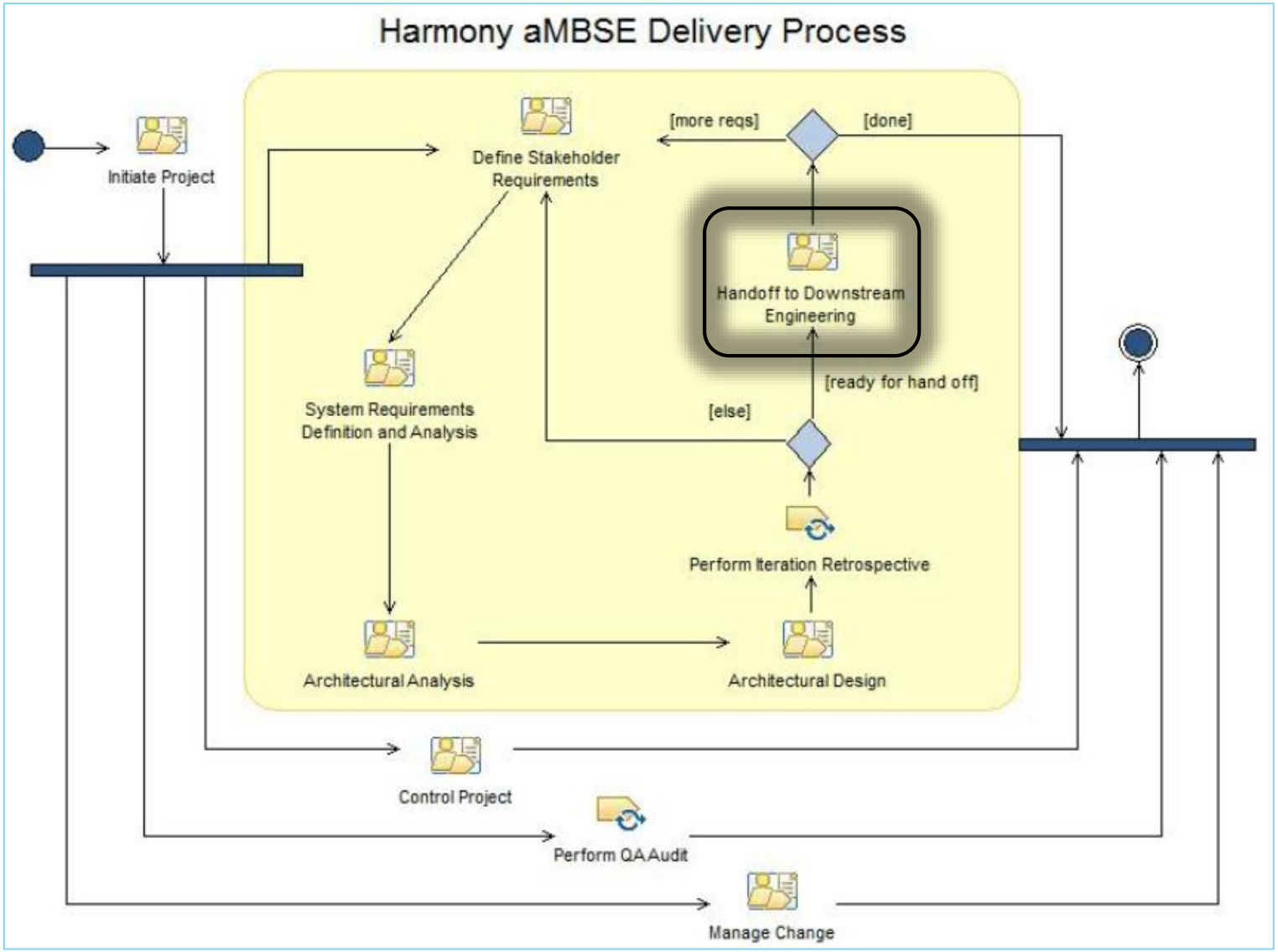
From Systems to Software: The Handoff

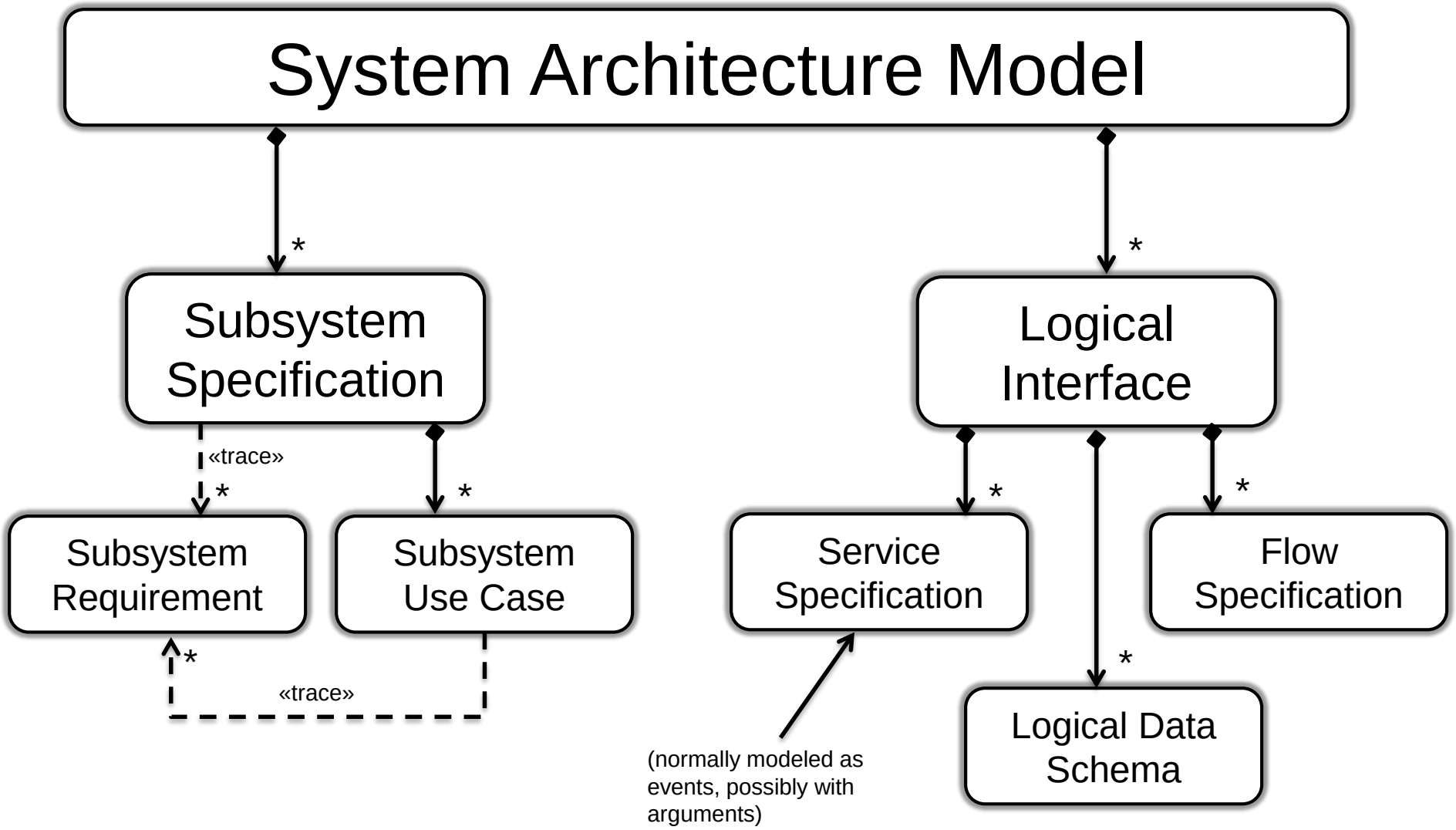
Bruce Powel Douglass, Ph.D.
Chief Evangelist,
IBM Internet of Things (IoT)
bruce.douglass@us.ibm.com
Twitter: @IronmanBruce
www.bruce-douglass.com



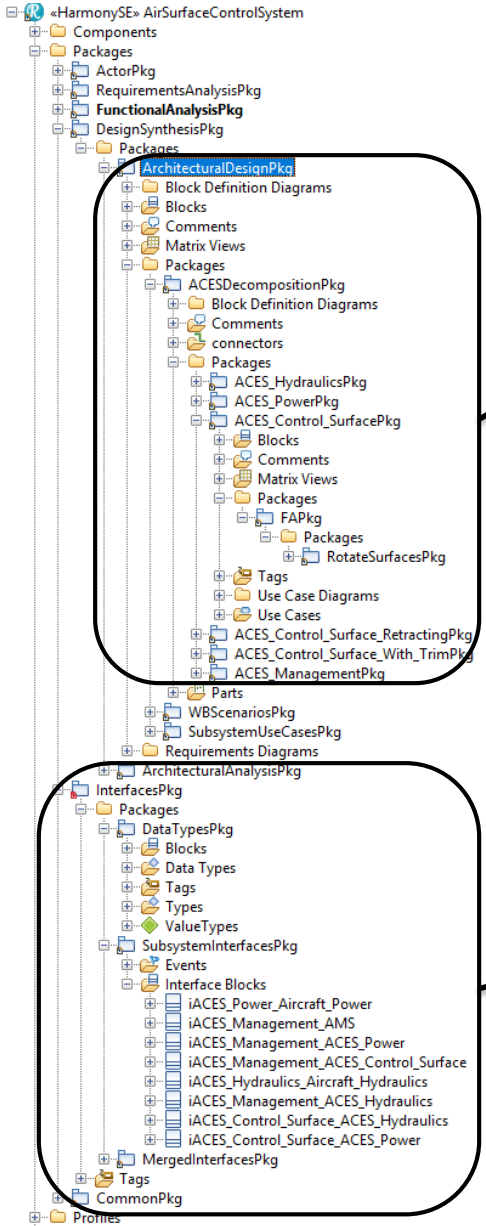
Harmony aMBSE Process

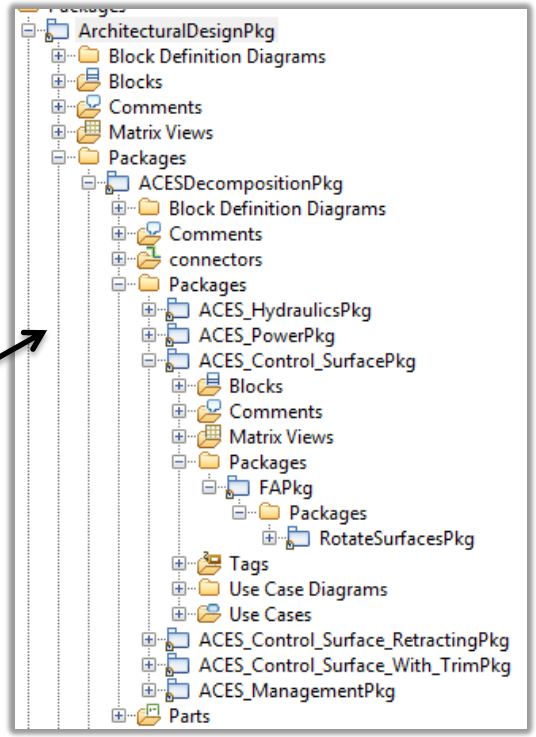


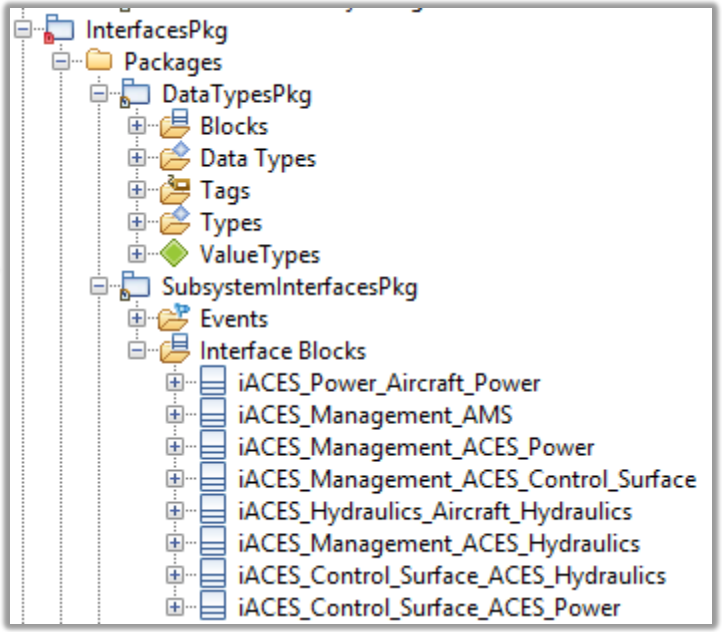
Handoff Prerequisites



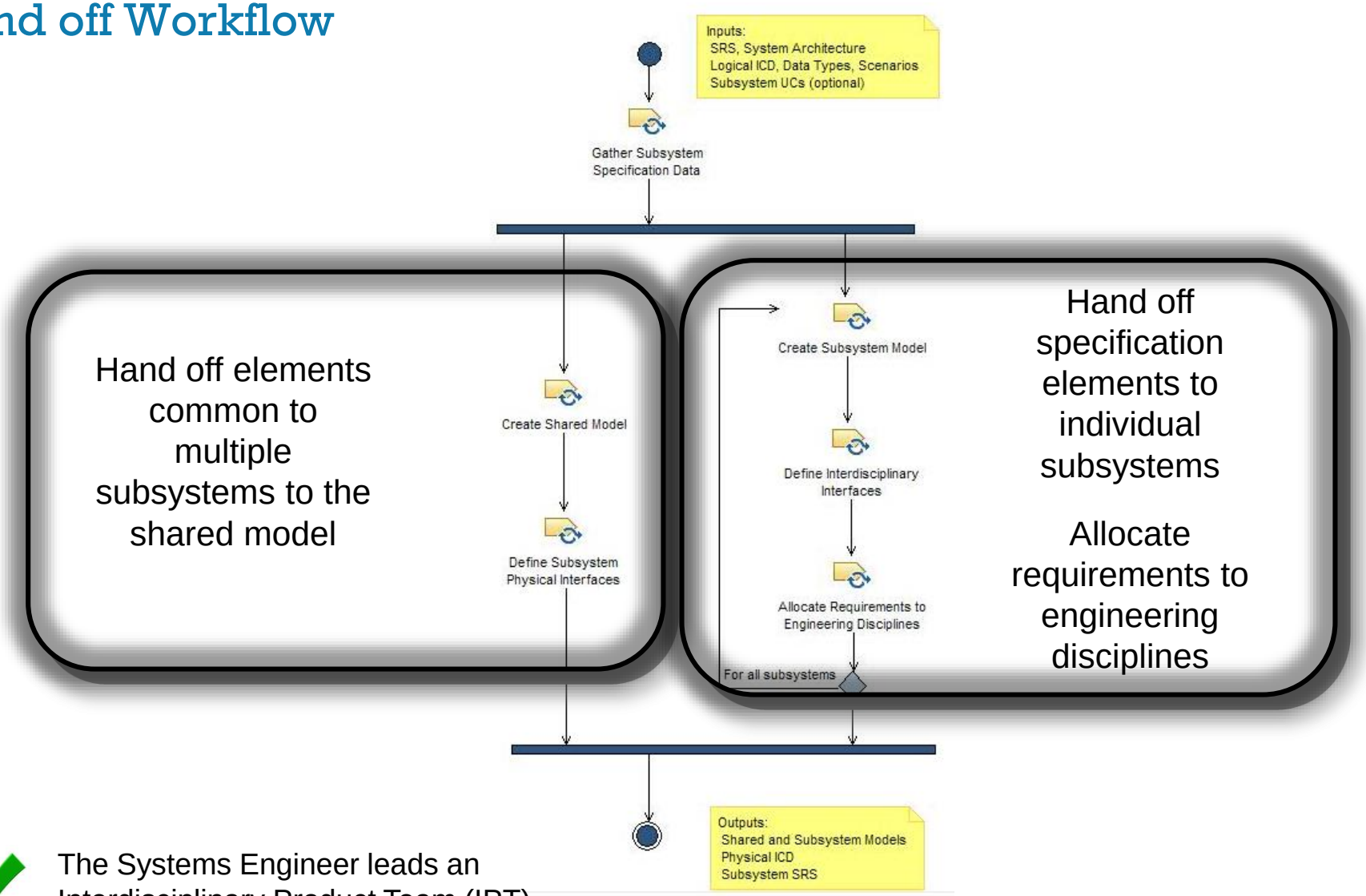
How is this Architectural Model Organized?







Hand off Workflow

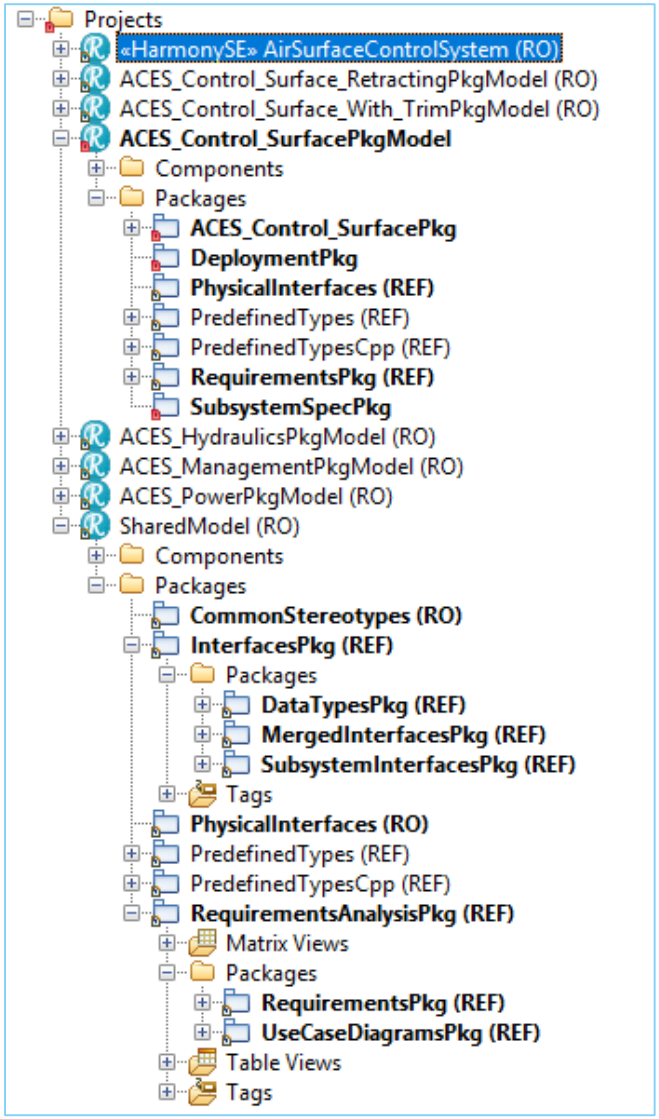


Useful Tips

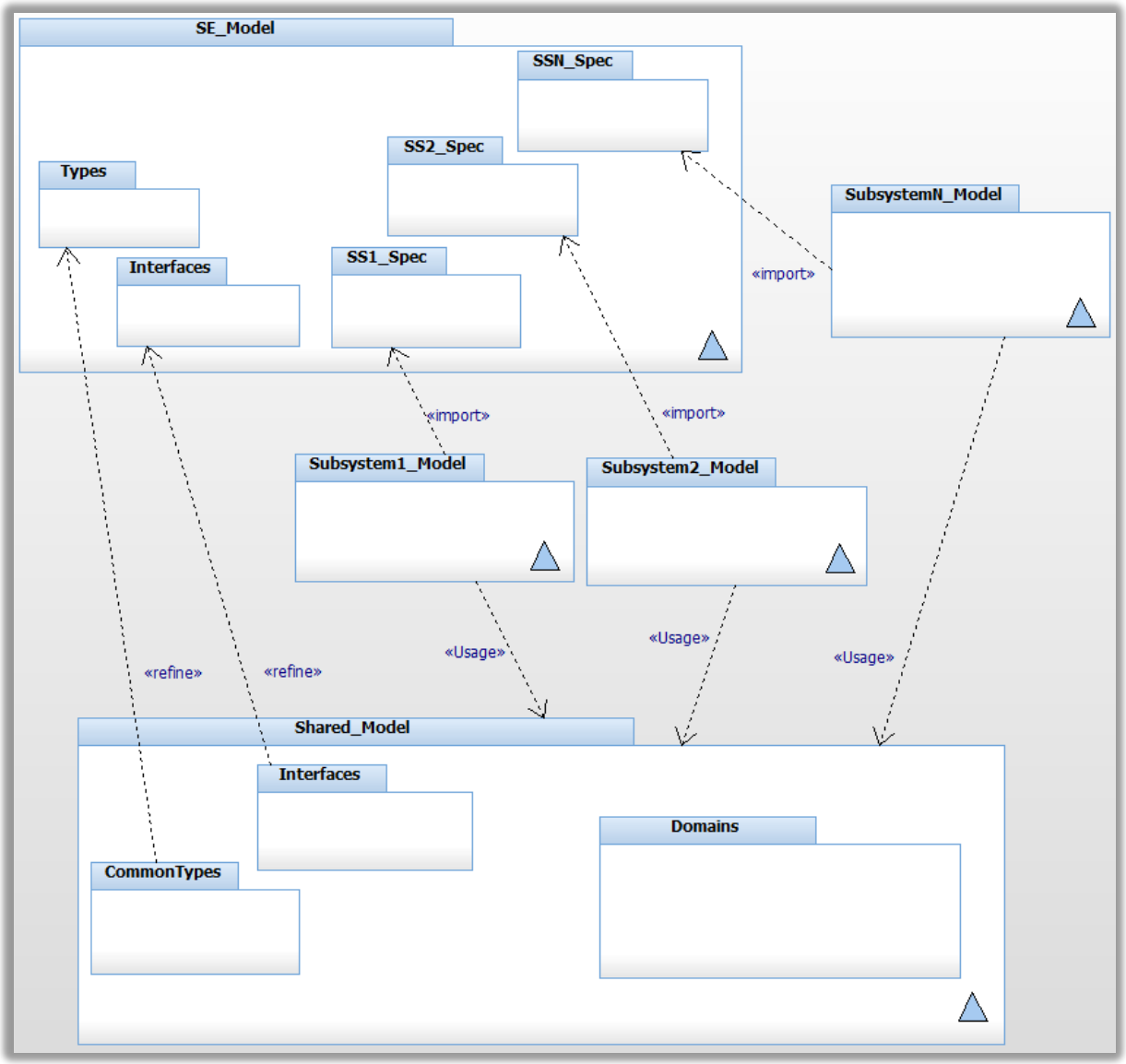
The Systems Engineer leads an Interdisciplinary Product Team (IPT) in the hand off effort.

Creation of Shared and Subsystem Models

- Either with the SE-Toolkit automation or manually, create
  - A singular Shared model for the physical interfaces and types they need
  - A separate model for each subsystem, which contains
    - (Copied) subsystem specification from the system engineering model
    - Reference to the requirements in the SE Model
    - Reference to the Shared Model

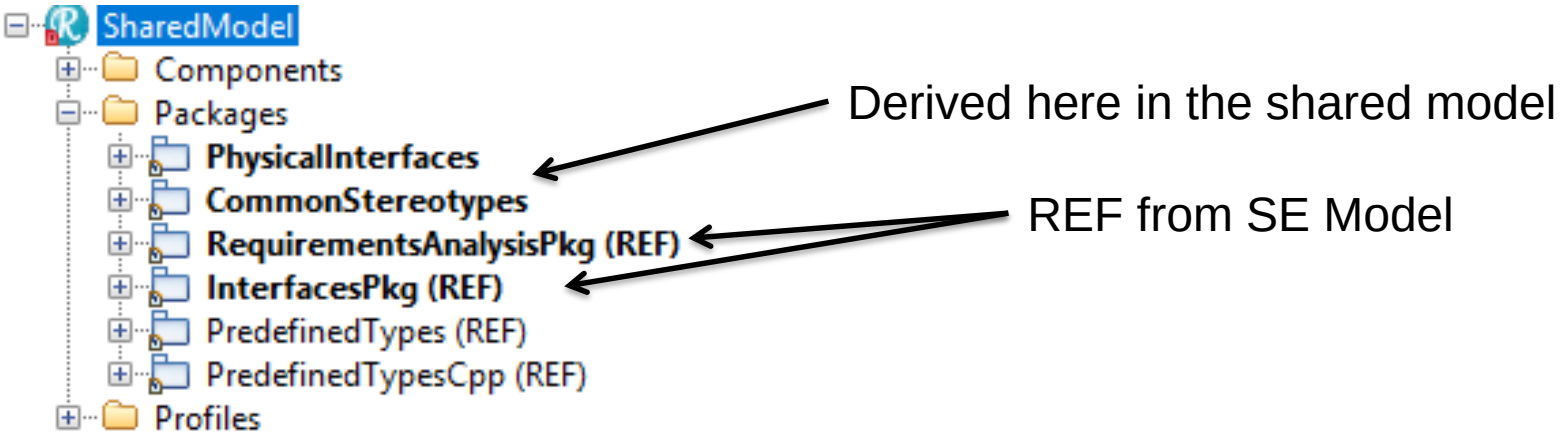


Canonical Model Organization



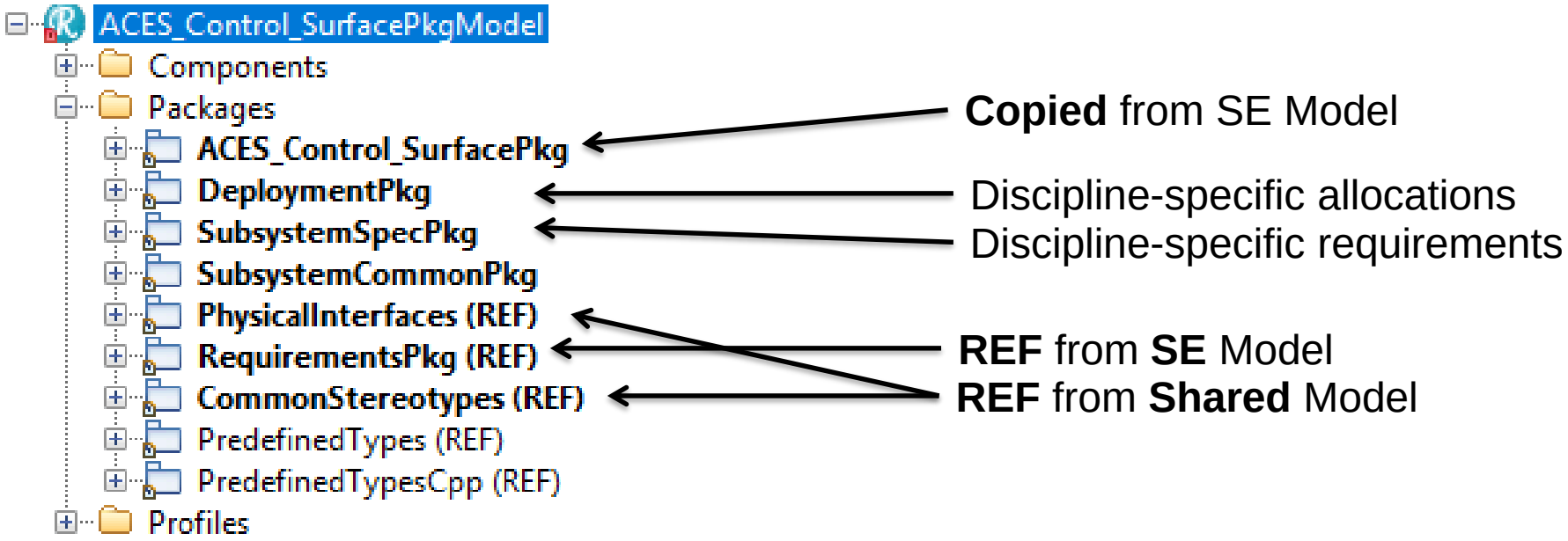
Shared Model Organization

- Purpose: To hold information relevant to more than one subsystem
  - Physical interfaces and physical data/flow schema
- Intent
  - Create the physical interfaces and trace them back to the logical interfaces
  - Create the physical data/flow schema and trace them back to the logical data/flows



Subsystem Model Organization

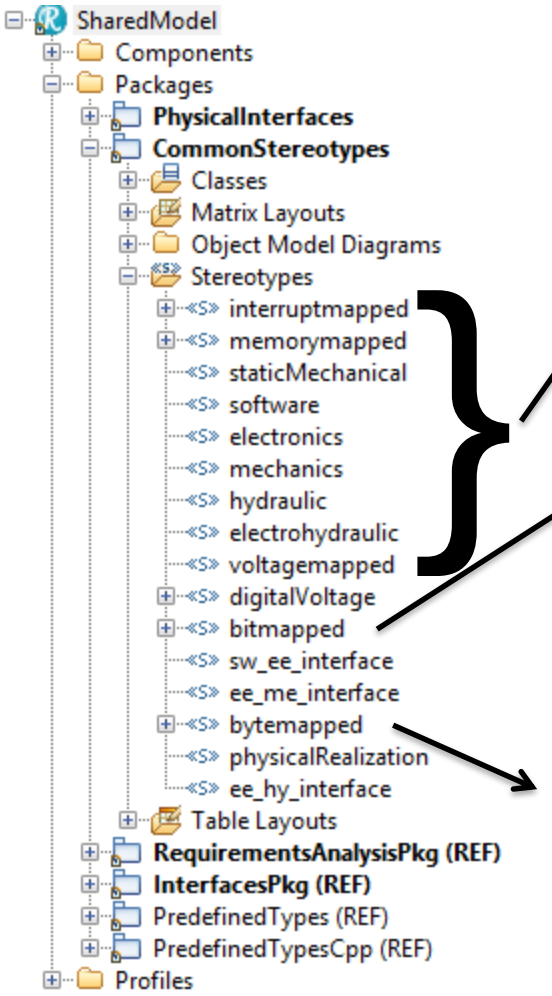
- Purpose: To hold information relevant to a single subsystem
  - Physical interfaces and physical data/flow schema
- Intent
  - Identify deployment facets, allocate requirements to them, and define their interfaces



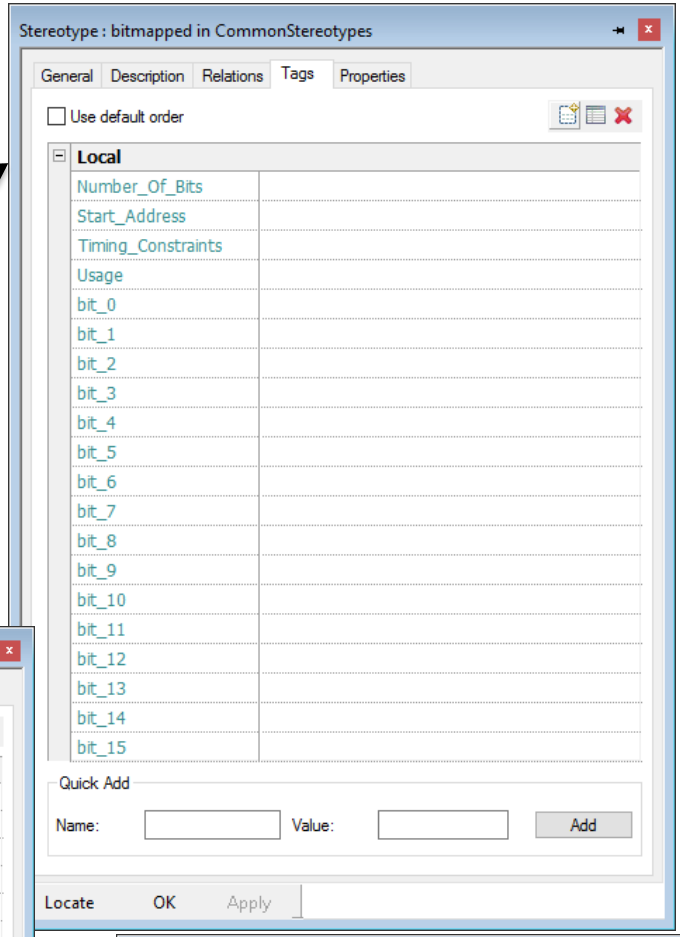
Modeling Physical Interfaces with Protocols

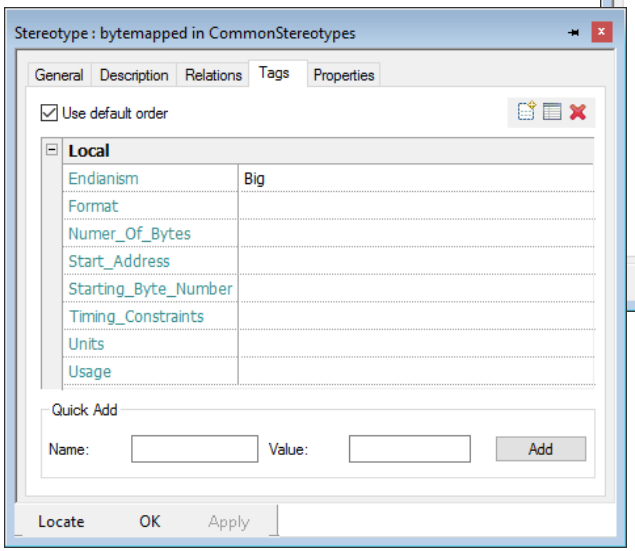
- Many interfaces will be implemented over a bus protocol:
  - Example
    - Logical: **Navigation** Subsystem ←→ **Radar** Subsystem uses event **herezaRadarTrack(t: RadarTrack)**
    - Physical: 1553 Bus Message with bit-level mapping to the message content
- Such interfaces are typically done as follows
  - Identify the protocol
    - If a standard – add a reference to the standard
    - If custom – define the protocol
  - Specify the *application-level protocol*
    - For every message in the logical interface, define a corresponding bus message
    - For every type in the logical interface, define the details of the corresponding bus message
  - Add **«represents»** dependency from the physical element to its corresponding logical element
  - Construct a table view
  - Review to ensure completeness, adequacy, correctness, and coverage

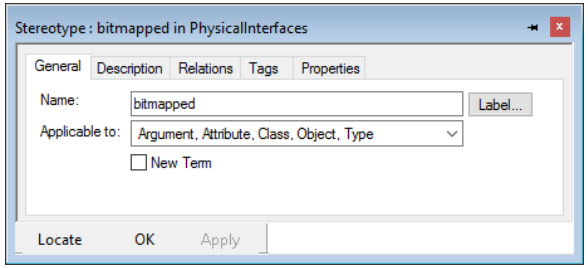
Useful Stereotypes for Physical Interfaces and Types



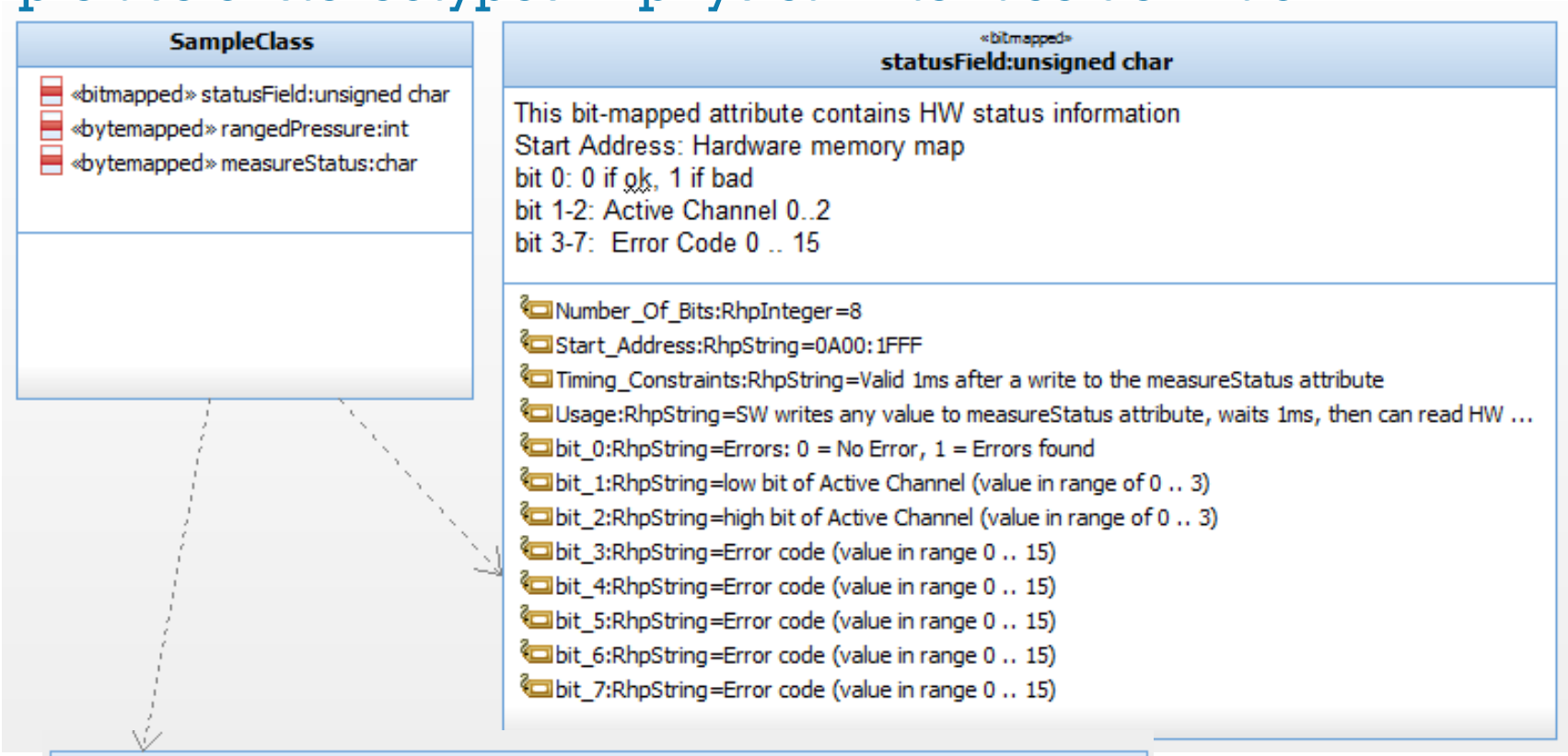
Used in deployment architecture to characterize engineering discipline-specific aspects (known as *deployment facets*)



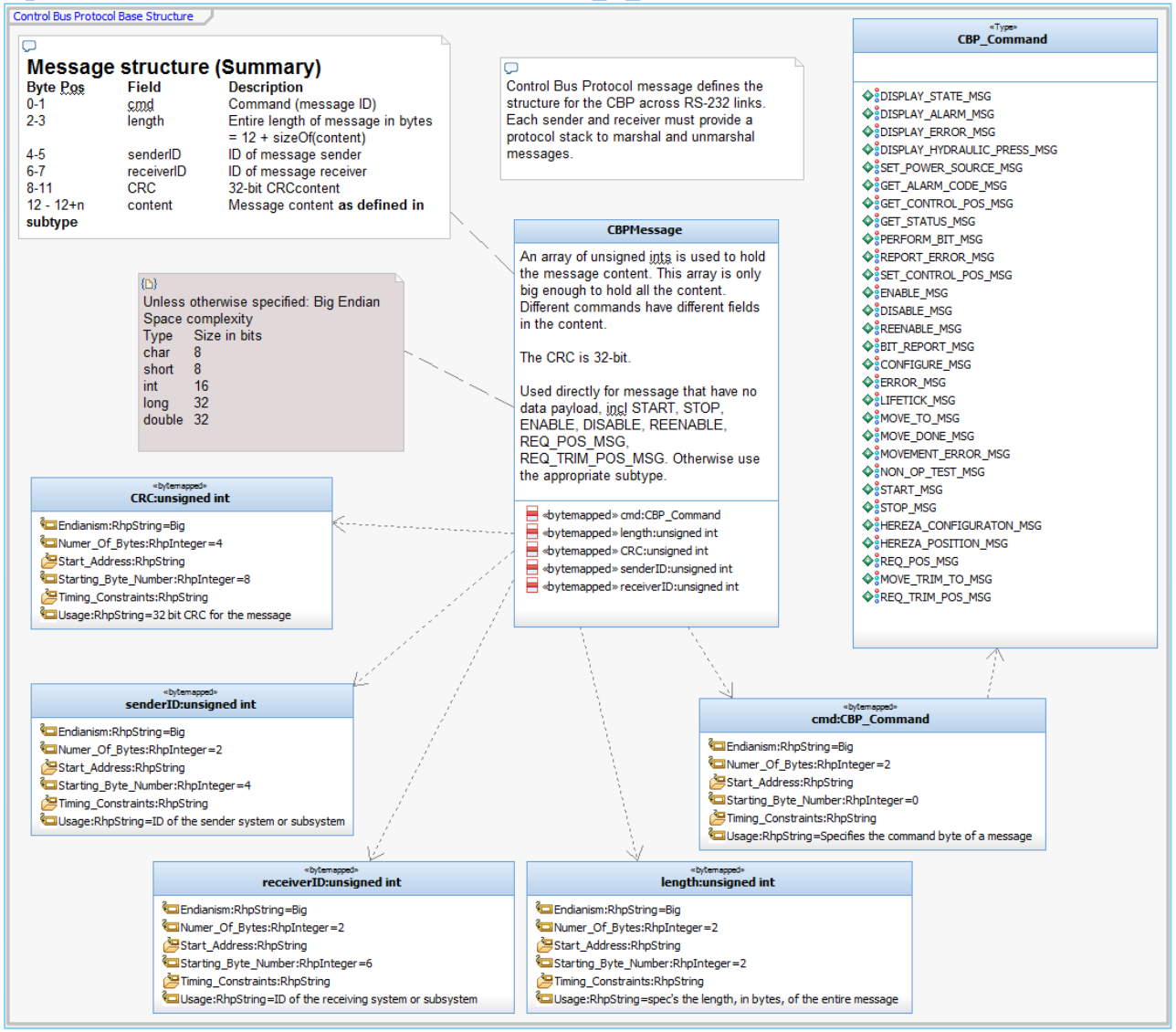




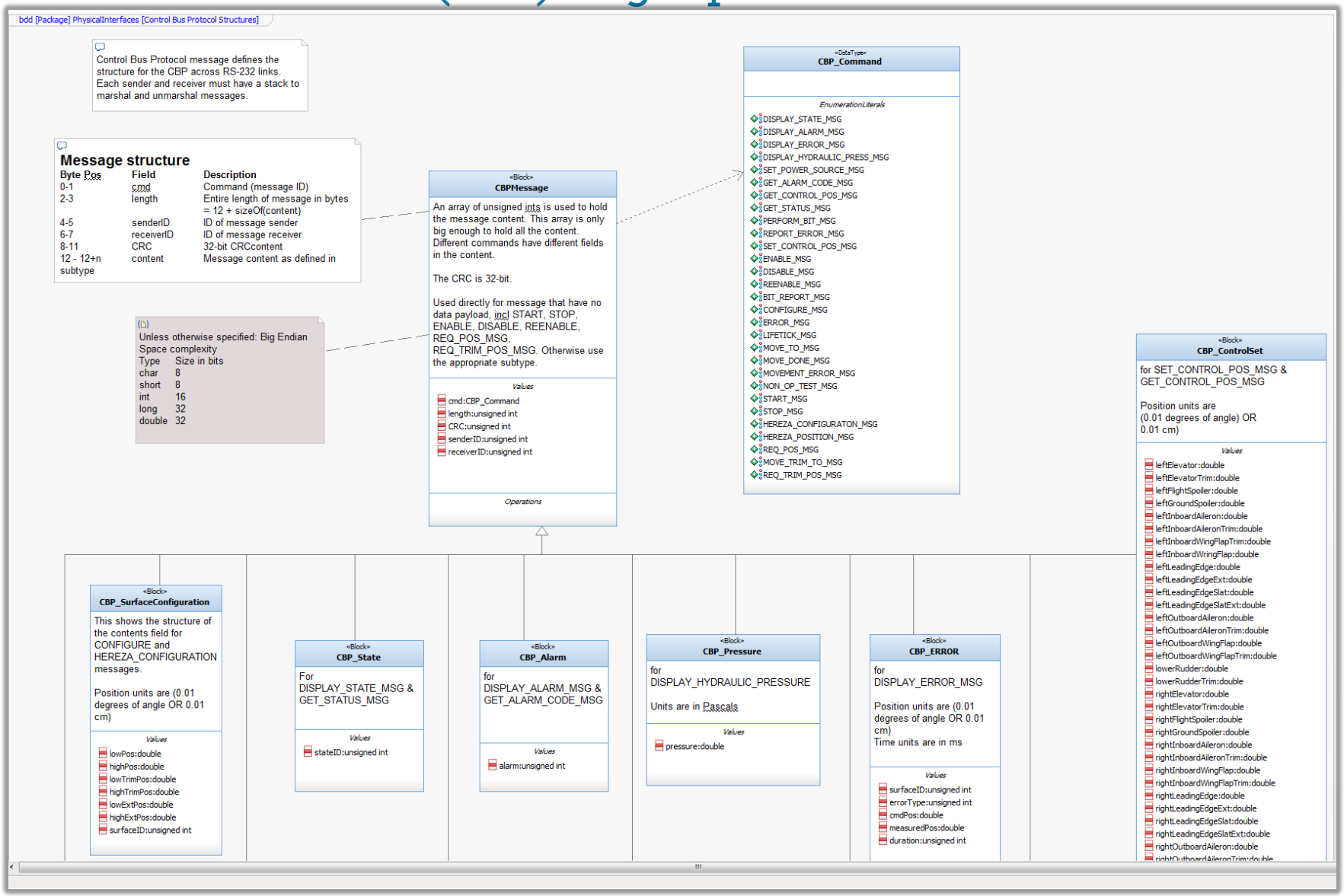
Example use of stereotypes in physical interface definition



Defining Physical Interfaces: The Application Protocol

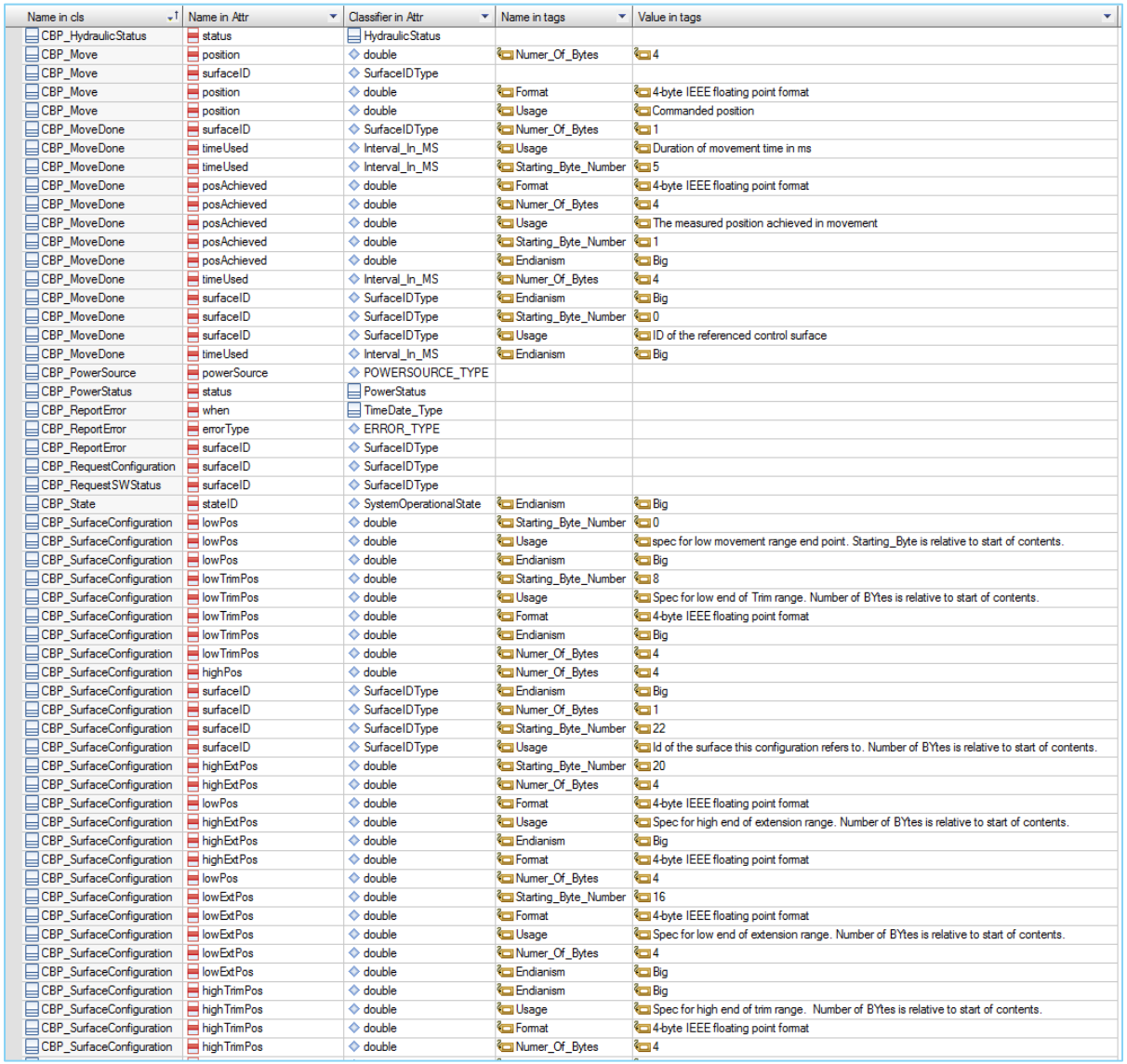


Control Bus Protocol (CBP) Msgs Spec



Showing the physical messaging details for an ICD*

- Rhapsody provides custom table layout tools using context patterns to show contents and metadata of the various messages. Here we see columns:
  - Message name
  - Message content field
  - Content field metadata
  - Content field metadata value

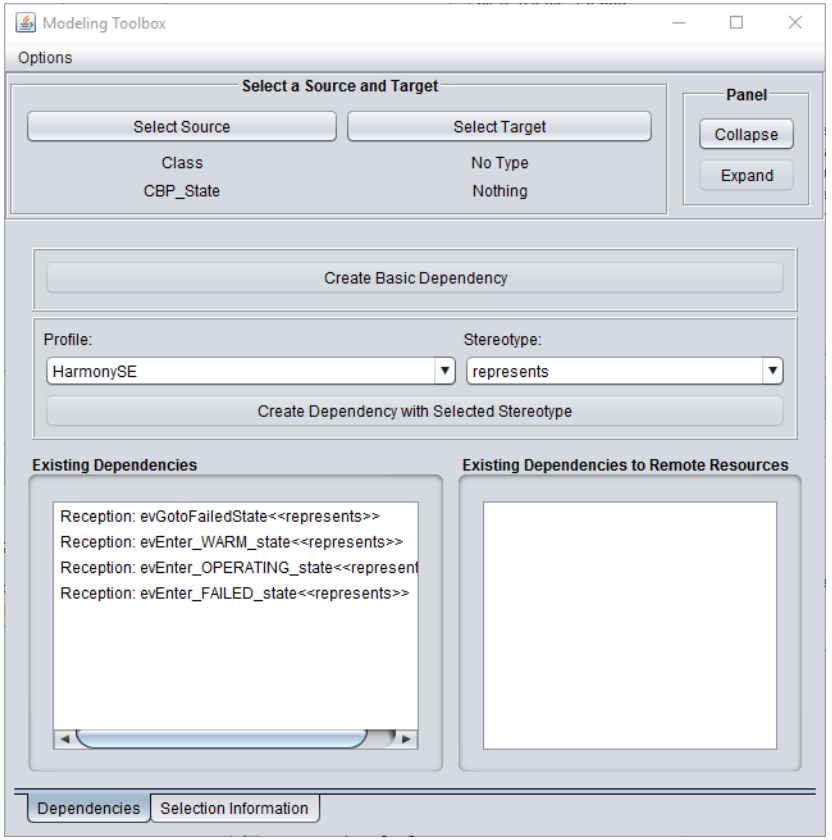


Name in cls	Name in Attr	Classifier in Attr	Name in tags	Value in tags
CBP_HydraulicStatus	status	HydraulicStatus		
CBP_Move	position	double	Numer_Of_Bytes	4
CBP_Move	surfaceID	SurfaceIDType		
CBP_Move	position	double	Format	4-byte IEEE floating point format
CBP_Move	position	double	Usage	Commanded position
CBP_MoveDone	surfaceID	SurfaceIDType	Numer_Of_Bytes	1
CBP_MoveDone	timeUsed	Interval_In_MS	Usage	Duration of movement time in ms
CBP_MoveDone	timeUsed	Interval_In_MS	Starting_Byte_Number	5
CBP_MoveDone	posAchieved	double	Format	4-byte IEEE floating point format
CBP_MoveDone	posAchieved	double	Numer_Of_Bytes	4
CBP_MoveDone	posAchieved	double	Usage	The measured position achieved in movement
CBP_MoveDone	posAchieved	double	Starting_Byte_Number	1
CBP_MoveDone	posAchieved	double	Endianism	Big
CBP_MoveDone	timeUsed	Interval_In_MS	Numer_Of_Bytes	4
CBP_MoveDone	surfaceID	SurfaceIDType	Endianism	Big
CBP_MoveDone	surfaceID	SurfaceIDType	Starting_Byte_Number	0
CBP_MoveDone	surfaceID	SurfaceIDType	Usage	ID of the referenced control surface
CBP_MoveDone	timeUsed	Interval_In_MS	Endianism	Big
CBP_PowerSource	powerSource	POWERSOURCE_TYPE		
CBP_PowerStatus	status	PowerStatus		
CBP_ReportError	when	TimeDate_Type		
CBP_ReportError	errorType	ERROR_TYPE		
CBP_ReportError	surfaceID	SurfaceIDType		
CBP_RequestConfiguration	surfaceID	SurfaceIDType		
CBP_RequestSWStatus	surfaceID	SurfaceIDType		
CBP_State	stateID	SystemOperationalState	Endianism	Big
CBP_SurfaceConfiguration	lowPos	double	Starting_Byte_Number	0
CBP_SurfaceConfiguration	lowPos	double	Usage	spec for low movement range end point. Starting_Byte is relative to start of contents.
CBP_SurfaceConfiguration	lowPos	double	Endianism	Big
CBP_SurfaceConfiguration	lowTrimPos	double	Starting_Byte_Number	8
CBP_SurfaceConfiguration	lowTrimPos	double	Usage	Spec for low end of Trim range. Number of BYtes is relative to start of contents.
CBP_SurfaceConfiguration	lowTrimPos	double	Format	4-byte IEEE floating point format
CBP_SurfaceConfiguration	lowTrimPos	double	Endianism	Big
CBP_SurfaceConfiguration	lowTrimPos	double	Numer_Of_Bytes	4
CBP_SurfaceConfiguration	highPos	double	Numer_Of_Bytes	4
CBP_SurfaceConfiguration	surfaceID	SurfaceIDType	Endianism	Big
CBP_SurfaceConfiguration	surfaceID	SurfaceIDType	Numer_Of_Bytes	1
CBP_SurfaceConfiguration	surfaceID	SurfaceIDType	Starting_Byte_Number	22
CBP_SurfaceConfiguration	surfaceID	SurfaceIDType	Usage	Id of the surface this configuration refers to. Number of BYtes is relative to start of contents.
CBP_SurfaceConfiguration	highExtPos	double	Starting_Byte_Number	20
CBP_SurfaceConfiguration	highExtPos	double	Numer_Of_Bytes	4
CBP_SurfaceConfiguration	lowPos	double	Format	4-byte IEEE floating point format
CBP_SurfaceConfiguration	highExtPos	double	Usage	Spec for high end of extension range. Number of BYtes is relative to start of contents.
CBP_SurfaceConfiguration	highExtPos	double	Endianism	Big
CBP_SurfaceConfiguration	highExtPos	double	Format	4-byte IEEE floating point format
CBP_SurfaceConfiguration	lowPos	double	Numer_Of_Bytes	4
CBP_SurfaceConfiguration	lowExtPos	double	Starting_Byte_Number	16
CBP_SurfaceConfiguration	lowExtPos	double	Format	4-byte IEEE floating point format
CBP_SurfaceConfiguration	lowExtPos	double	Usage	Spec for low end of extension range. Number of BYtes is relative to start of contents.
CBP_SurfaceConfiguration	lowExtPos	double	Numer_Of_Bytes	4
CBP_SurfaceConfiguration	lowExtPos	double	Endianism	Big
CBP_SurfaceConfiguration	highTrimPos	double	Endianism	Big
CBP_SurfaceConfiguration	highTrimPos	double	Usage	Spec for high end of trim range. Number of BYtes is relative to start of contents.
CBP_SurfaceConfiguration	highTrimPos	double	Format	4-byte IEEE floating point format
CBP_SurfaceConfiguration	highTrimPos	double	Numer_Of_Bytes	4

* Interface Control Document

Adding traceability between logical and physical interfaces

- Harmony SE has a **«represents»** stereotype to traces elements across different abstraction levels
- This can be done diagrammatically or using the Harmony SE Modeling Toolkit
- For every physical interface service
  - Create a **«represents»** dependency back to the logical interface element it represents

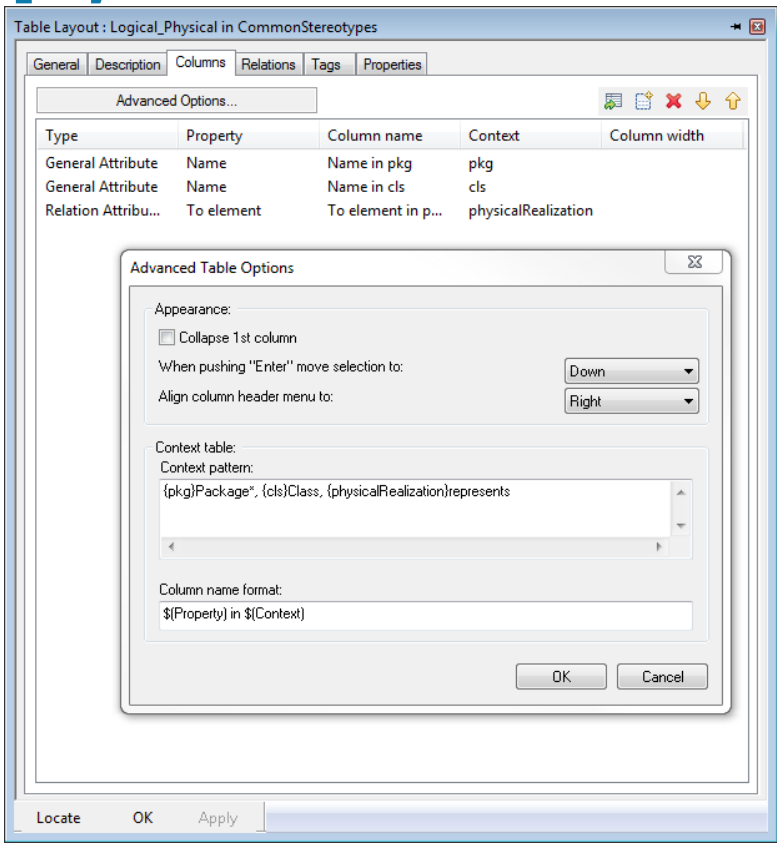


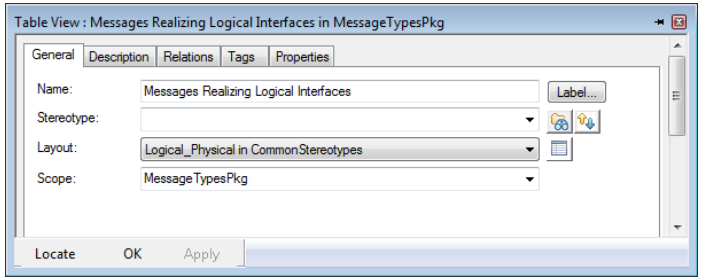
Adding traceability between logical and physical interfaces

☑ Create a table to view the relations

Found 28 elements

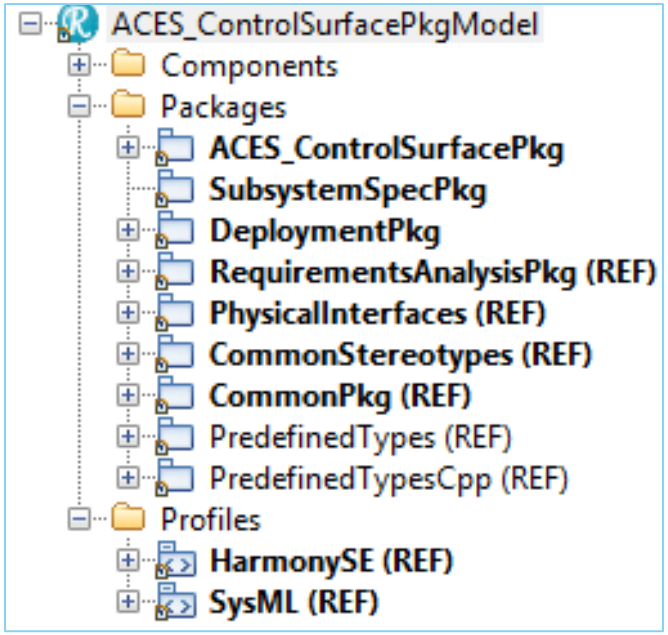
Name in pkg	Name in PhysicalRepresentation	To element in LogicalRepresenta...
MessageTypesPkg		
	CBPMessage	checkPower
	CBPMessage	evRequestHydraulicStatus
	CBPMessage	reqENABLE_Command
	CBPMessage	request_Hydraulic_Status
	CBP_ControlSet	herezaPositionSet
	CBP_Error	evError
	CBP_HydraulicStatus	herezaHydraulic_Pressure
	CBP_HydraulicStatus	herezaHydraulic_Pressure
	CBP_Move	Command_To_Position
	CBP_Move	evMoveTo
	CBP_MoveDone	Updated_Position
	CBP_MoveDone	evMovementDone
	CBP_MoveDone	herezaPosition
	CBP_PowerSource	Select_Battery_As_Source
	CBP_PowerStatus	Update_Power_Status
	CBP_PowerStatus	updatePowerStatus
	CBP_ReportError	ReportError
	CBP_RequestConfiguration	Req_config_parameter
	CBP_RequestConfiguration	Req_config_parameters
	CBP_RequestSWStatus	request_SW_Integrity_Check
	CBP_RequestSWStatus	request_SW_Integrity_Check
	CBP_SWStatus	SW_Status
	CBP_SWStatus	SW_Status
	CBP_State	evEnter_FAILED_state
	CBP_State	evEnter_OPERATING_state
	CBP_State	evEnter_WARM_state
	CBP_State	evGotoFailedState
	CBP_SurfaceConfiguration	herezaConfiguration





Subsystem models

- There is a separate package created for each subsystem model

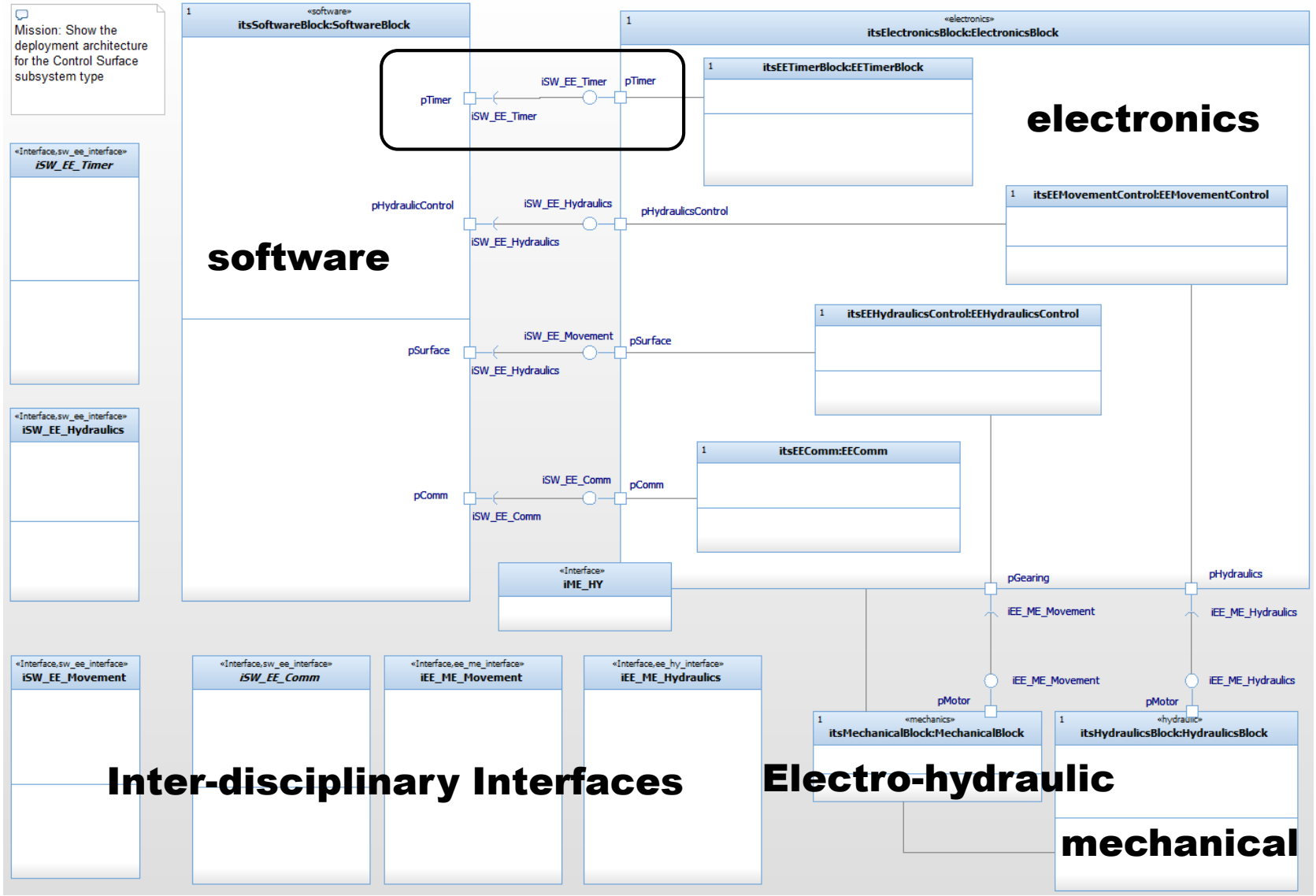


- We will
  - ✓ Identify the engineering disciplines involved
  - ✓ Allocate requirements to those engineering disciplines
  - ✓ Specify the interfaces between those engineering disciplines

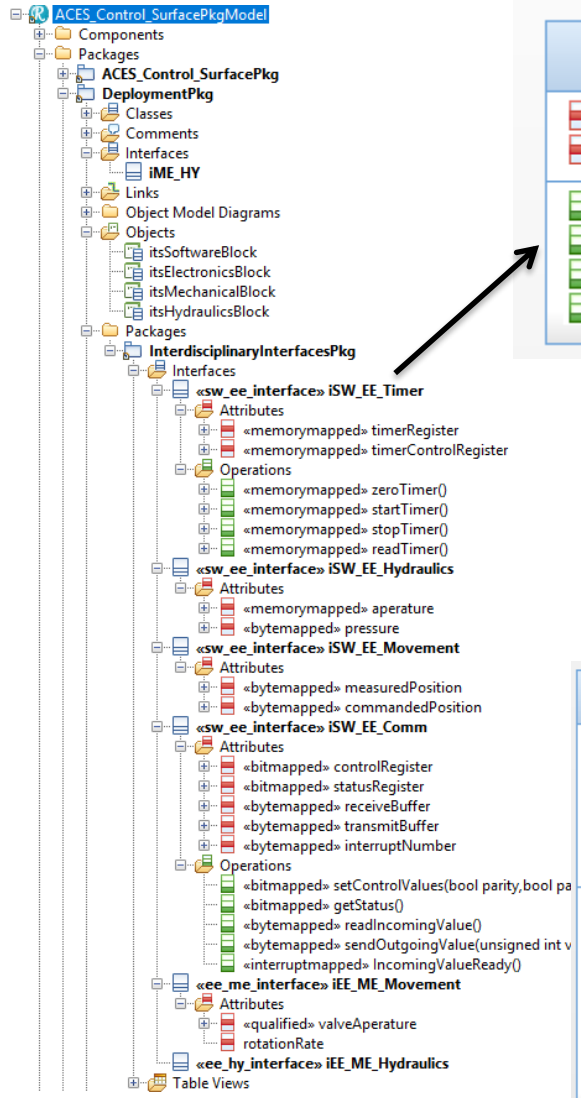
Deployment Architecture

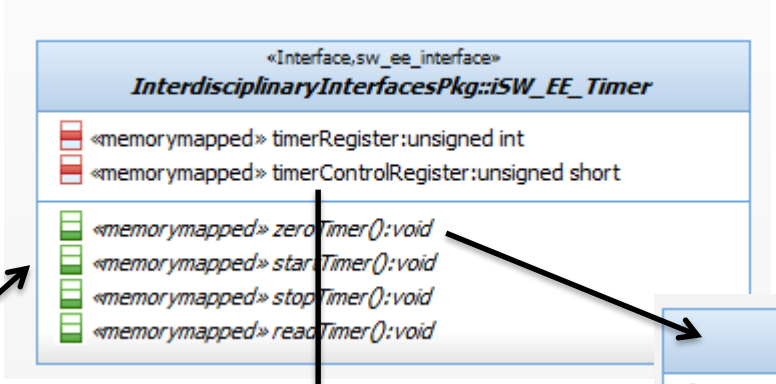
- The Harmony Process defines the deployment architecture:
  - The *Deployment Architecture* is the
    - Identification of ***Deployment Facets*** (architectural elements from different engineering disciplines)
    - The assignment of functional and other responsibilities to those deployment facets
    - The definition of interfaces between deployment facets from different engineering disciplines
- Deployment architecture specifically *does not* identify discipline-specific architecture within the deployment facets
  - The internal structuring of deployment facets is the responsibility of the related engineering discipline
- Deployment architecture is shown on block (class) diagrams with
  - Stereotyped deployment elements indicating their pedigree
  - Interfaces between the deployment elements
- Harmony uses
  - *Ports and connectors* for dynamic connections; e.g. those that carry information, energy, materiel or fluids
  - *Associations and links* for static connections; e.g. those such as mechanical fastenings and cable management

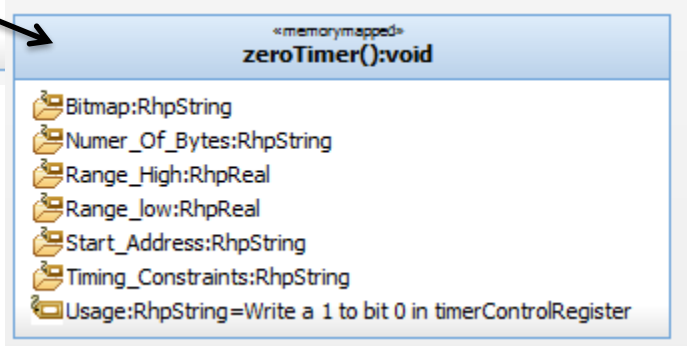
Deployment Architecture Facets for Control Air Surfaces Subsystem

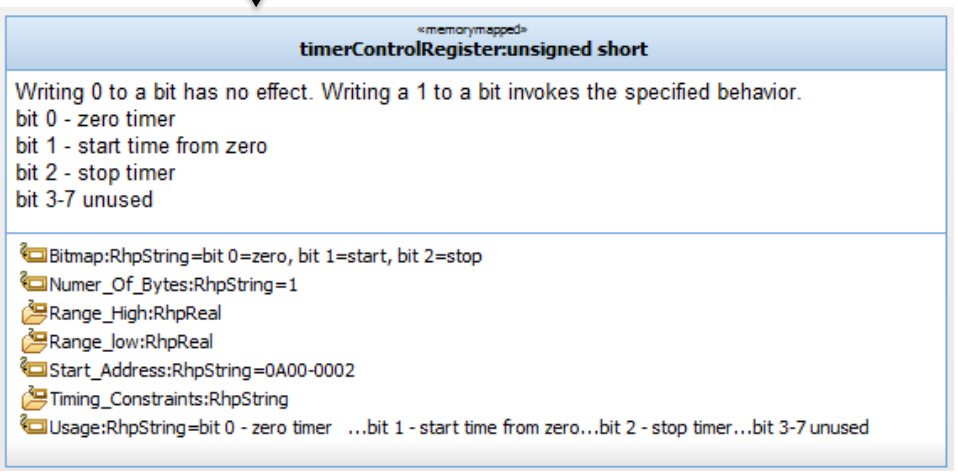


Define the Interdisciplinary Interfaces









Show the Interdisciplinary Interface Details in a table

Class	Operation	Attribute	Attribute Type	Tag	Tag Value
iEE_ME_Hydraulics					
iEE_ME_Movement					
		rotationRate	int		
		valveAperature	double	High_Range	2.5
		valveAperature	double	Low_Range	0
		valveAperature	double	Units	cm
		valveAperature	double	accuracy	0.01
		valveAperature	double	precision	0.01
iSW_EE_Comm					
		controlRegister	unsigned int	Number_Of_Bits	5
		controlRegister	unsigned int	Start_Address	0A00:0100
		controlRegister	unsigned int	Usage	Sets comm parameters
		controlRegister	unsigned int	bit_0	Parity: Disable (0) Enable (1)
		controlRegister	unsigned int	bit_1	Parity Mode: Even (0), Odd (1)
		controlRegister	unsigned int	bit_10	InterruptEnable: Disable (0), Enable (1)
		controlRegister	unsigned int	bit_2	LSB First (0), MSB First (1)
		controlRegister	unsigned int	bit_3	Data Length: 7-bit (0), 8-bit (1)
		controlRegister	unsigned int	bit_4	Number of Stop bits: One stop bit (0), two stop bits (1)
		controlRegister	unsigned int	bit_5	Channel number bit 0
		controlRegister	unsigned int	bit_6	Channel number bit 1
		controlRegister	unsigned int	bit_7	Channel number bit 2
		controlRegister	unsigned int	bit_8	Channel number bit 3
		interruptNumber	unsigned int	Numer_Of_Bytes	2
		interruptNumber	unsigned int	Start_Address	0A00:0106
		interruptNumber	unsigned int	Usage	This specifies the software interrupt to raise when data arrives in the receive buffer and interrupts are enabled in the
		receiveBuffer	unsigned int	Numer_Of_Bytes	2
		receiveBuffer	unsigned int	Start_Address	0A00:0A10
		statusRegister	unsigned int	Start_Address	0A00:0102
		statusRegister	unsigned int	Usage	Provides hw comm status
		statusRegister	unsigned int	bit_0	Loopback (Listen enable) 0=disable, 1=enable
		statusRegister	unsigned int	bit_1	Framing error detected 0=No error, 1=error
		statusRegister	unsigned int	bit_10	
		statusRegister	unsigned int	bit_2	Overrun error detected 0=No error, 1=error
		statusRegister	unsigned int	bit_3	Parity error detected 0=No error, 1=error
		transmitBuffer	unsigned int	Numer_Of_Bytes	2
		transmitBuffer	unsigned int	Start_Address	0A00:0104
		transmitBuffer	unsigned int	Usage	A write to this address signals the system to send the 2 bytes to the selected
	IncomingValueReady				
	getStatus				
	readIncomingValue				
	sendOutgoingValue				
	setControlValues	receiveBuffer	unsigned int	Usage	Read incoming data as 2 bytes
iSW_EE_Hydraulics					
		aperature	unsigned int	Numer_Of_Bytes	2
		aperature	unsigned int	Range_High	65535 (fully open)
		aperature	unsigned int	Range_low	0 (fully closed)
		aperature	unsigned int	Start_Address	0A00:0120
		aperature	unsigned int	Usage	R/W. Write to set the aperature. Read to see it's current value
		pressure	int	Numer_Of_Bytes	2
		pressure	int	Start_Address	0A00:0122
		pressure	int	Units	kiloPascals (kPa)
		pressure	int	Usage	Read only. Provides the current value of pressure in the hydraulic circuit.

Class	Operation	Attribute	Attribute Type	Tag	Tag Value
iSW_EE_Movement					
		commandedPositi...	int	Numer_Of_Bytes	2
		commandedPositi...	int	Start_Address	0A00:0126
		commandedPositi...	int	Units	0.01 Degrees
		commandedPositi...	int	Usage	Sets the commanded angle from -180 to 180 degrees in units of 0.01 degrees.
		measuredPosition	int	Numer_Of_Bytes	2
		measuredPosition	int	Start_Address	0A00:0124
		measuredPosition	int	Units	0.01 Degrees.
		measuredPosition	int	Usage	Read only. Provide measured position in range -180 to 180 in 0.01Degree units
iSW_EE_Timer					
		timerControlRegis...	unsigned short	Bitmap	bit 0=zero, bit 1=start, bit 2=stop
		timerControlRegis...	unsigned short	Numer_Of_Bytes	1
		timerControlRegis...	unsigned short	Start_Address	0A00-0002
		timerControlRegis...	unsigned short	Usage	bit 0 - zero timer bit 1 - start time from zero bit 2 - stop timer bit 3-7 unused
		timerRegister	unsigned int	Bitmap	32 bit unsigned
		timerRegister	unsigned int	Numer_Of_Bytes	4
		timerRegister	unsigned int	Range_High	2^32 - 1
		timerRegister	unsigned int	Range_low	0
		timerRegister	unsigned int	Start_Address	0A00-0000
		timerRegister	unsigned int	Usage	This is a read only register holding the elapsed time in ms
	readTimer	timerRegister	unsigned int	Usage	First stop the time by writing 1 to bit 2. Then read the value (in ms) from the
	startTimer			Usage	Write a 1 bit to bit 1. Starts time from current value. If a 1 bit iis written to bit 0, the timer starts from 0
	stopTimer			Usage	Write a 1 bit to bit 2. Stops the timer.
	zeroTimer			Usage	Write a 1 to bit 0 in timerControlRegister

- For each interface:
  - Operation
    - Metadata attributes
  - Attribute / value property
    - Metadata attributes

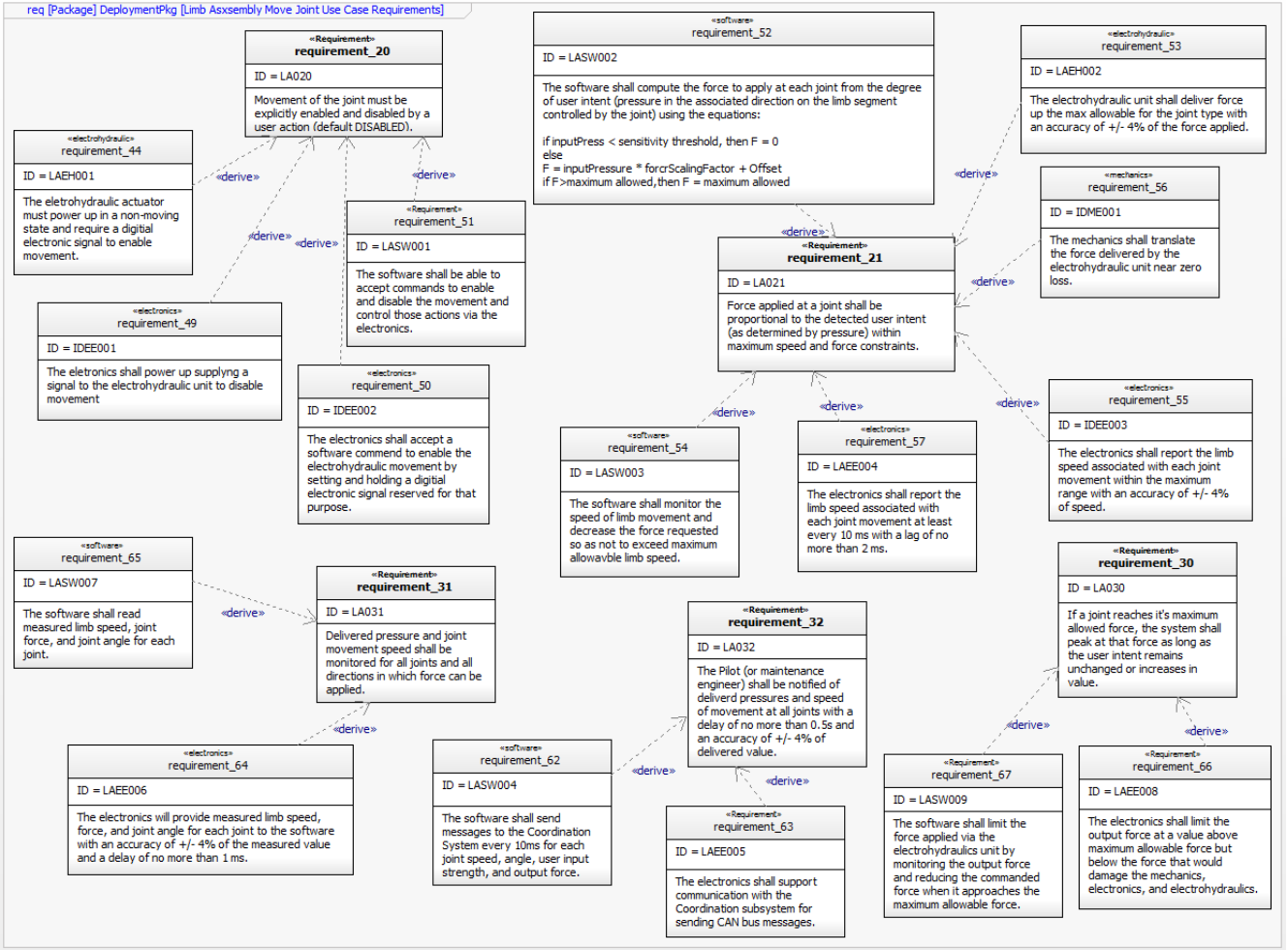
Derivation for Deployment Facets Requirements

- While some requirements may be directly allocated to a deployment facet, many must be decomposed into derived requirements before allocation.

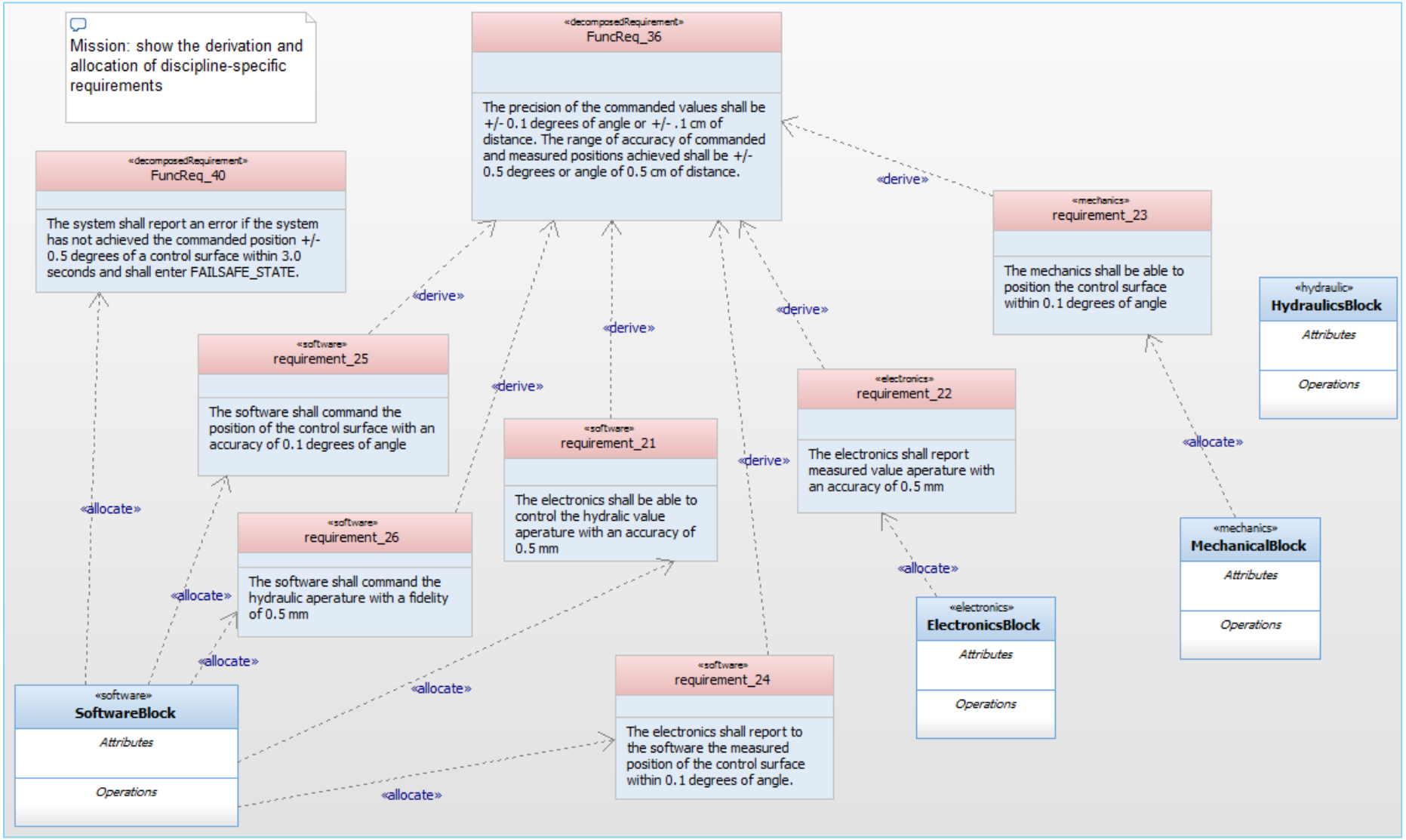
electrohydraulics

electronics

software



Allocations of Requirements to Deployment Facets



Allocations of Requirements to Deployment Facets

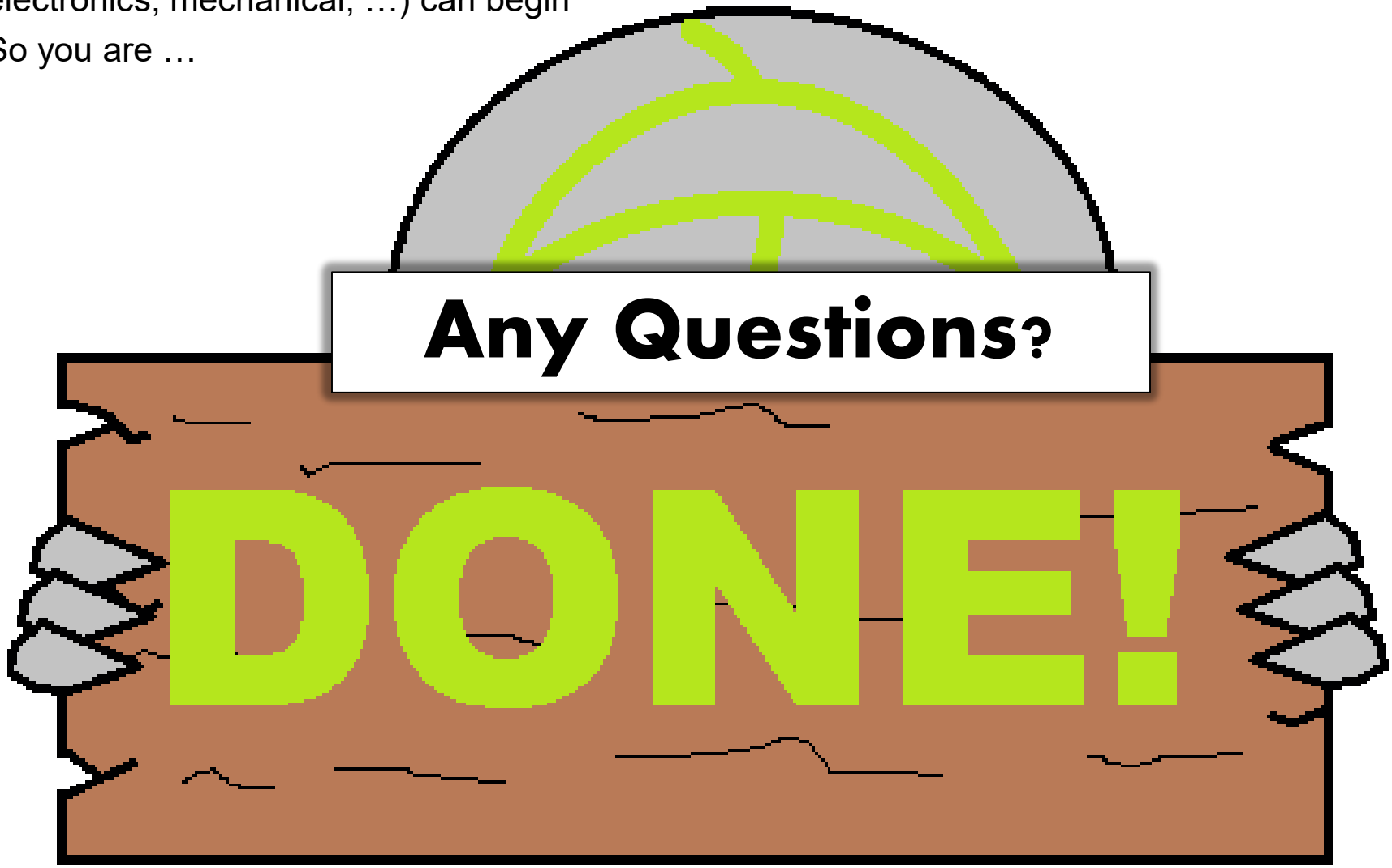
Requirement Subsystem Derived Requirement

Name	From	To	Description
ACES_SS_requirement_32	ACES_ControlSurface	ACES_SS_requirement_32	
ACES_SS_requirement_33	ACES_ControlSurface	ACES_SS_requirement_33	
ACES_SS_requirement_34	ACES_ControlSurface	ACES_SS_requirement_34	
ACSCUNT_requirement_10	ACES_ControlSurface	ACSCUNT_requirement_10	
ACSCUNT_requirement_11	ACES_ControlSurface	ACSCUNT_requirement_11	
ACSCUNT_requirement_12	ACES_ControlSurface	ACSCUNT_requirement_12	
ACSCUNT_requirement_13	ACES_ControlSurface	ACSCUNT_requirement_13	
ACSCUNT_requirement_18	ACES_ControlSurface	ACSCUNT_requirement_18	
ACSCUNT_requirement_19	ACES_ControlSurface	ACSCUNT_requirement_19	
ACSCUNT_requirement_21	ACES_ControlSurface	ACSCUNT_requirement_21	
ACSCUNT_requirement_24	ACES_ControlSurface	ACSCUNT_requirement_24	
ACSCUNT_requirement_25	ACES_ControlSurface	ACSCUNT_requirement_25	
ACSCUNT_requirement_26	ACES_ControlSurface	ACSCUNT_requirement_26	
ACSCUNT_requirement_3	ACES_ControlSurface	ACSCUNT_requirement_3	
ACSCUNT_requirement_7	ACES_ControlSurface	ACSCUNT_requirement_7	
ConfigReq_1	ACES_ControlSurface	ConfigReq_1	
ConfigReq_3	ACES_ControlSurface	ConfigReq_3	
DerConfigReq_1	ACES_ControlSurface	DerConfigReq_1	
DerConfigReq_2	ACES_ControlSurface	DerConfigReq_2	
DerFunReq_1	ACES_ControlSurface	DerFunReq_1	
DerIntReq_1	ACES_ControlSurface	DerIntReq_1	
DerIntReq_10	ACES_ControlSurface	DerIntReq_10	
DerIntReq_11	ACES_ControlSurface	DerIntReq_11	
DerIntReq_12	ACES_ControlSurface	DerIntReq_12	
DerIntReq_14	ACES_ControlSurface	DerIntReq_14	
DerIntReq_2	ACES_ControlSurface	DerIntReq_2	
DerIntReq_8	ACES_ControlSurface	DerIntReq_8	
DerReqInt_13	ACES_ControlSurface	DerReqInt_13	
DerStartUpReq_1	ACES_ControlSurface	DerStartUpReq_1	
DerStartupReq_2	ACES_ControlSurface	DerStartupReq_2	
DerStartupReq_3	ACES_ControlSurface	DerStartupReq_3	
ErrorReq_26	ACES_ControlSurface	ErrorReq_26	
ErrorReq_27	ACES_ControlSurface	ErrorReq_27	
ErrorReq_28	ACES_ControlSurface	ErrorReq_28	
ErrorReq_29	ACES_ControlSurface	ErrorReq_29	
ErrorReq_3	ACES_ControlSurface	ErrorReq_3	
ErrorReq_34	ACES_ControlSurface	ErrorReq_34	
ErrorReq_35	ACES_ControlSurface	ErrorReq_35	
ErrorReq_36	ACES_ControlSurface	ErrorReq_36	
ErrorReq_37	ACES_ControlSurface	ErrorReq_37	

Name	From	To	Description
FuncReq_25	ACES_ControlSurface	FuncReq_25	
FuncReq_27	ACES_ControlSurface	FuncReq_27	
FuncReq_28	ACES_ControlSurface	FuncReq_28	
FuncReq_29	ACES_ControlSurface	FuncReq_29	
FuncReq_30	ACES_ControlSurface	FuncReq_30	
FuncReq_36	ACES_ControlSurface	FuncReq_36	
FuncReq_37	ACES_ControlSurface	FuncReq_37	
FuncReq_40	ACES_ControlSurface	FuncReq_40	
FuncReq_40	SoftwareBlock	FuncReq_40	
OtherReq_0	ACES_ControlSurface	OtherReq_0	
OtherReq_1	ACES_ControlSurface	OtherReq_1	
requirement_13	SoftwareBlock	requirement_13	
requirement_14	SoftwareBlock	requirement_14	
requirement_15	SoftwareBlock	requirement_15	
requirement_16	ElectronicsBlock	requirement_16	
requirement_17	MechanicalBlock	requirement_17	
requirement_18	HydraulicsBlock	requirement_18	
requirement_21	SoftwareBlock	requirement_21	
requirement_22	ElectronicsBlock	requirement_22	
requirement_23	MechanicalBlock	requirement_23	
requirement_24	SoftwareBlock	requirement_24	
requirement_25	SoftwareBlock	requirement_25	
requirement_26	SoftwareBlock	requirement_26	
requirement_28	SoftwareBlock	requirement_28	
requirement_30	ElectronicsBlock	requirement_30	
requirement_31	MechanicalBlock	requirement_31	
requirement_34	ElectronicsBlock	requirement_34	
requirement_35	MechanicalBlock	requirement_35	
requirement_36	HydraulicsBlock	requirement_36	
requirement_37	SoftwareBlock	requirement_37	
requirement_38	SoftwareBlock	requirement_38	
requirement_41	SoftwareBlock	requirement_41	
requirement_42	SoftwareBlock	requirement_42	
requirement_43	ElectronicsBlock	requirement_43	
requirement_44	ElectronicsBlock	requirement_44	
requirement_45	SoftwareBlock	requirement_45	
requirement_46	ElectronicsBlock	requirement_46	
requirement_47	HydraulicsBlock	requirement_47	
requirement_48	MechanicalBlock	requirement_48	

Hand off from Systems to Software

- At this point, the hand off is complete, and the down stream engineering (software, electronics, mechanical, …) can begin
- So you are …



Real-Time Agile Systems and Software Development

Welcome to **www.bruce-douglass.com**

You've found yourself on **www.bruce-douglass.com,** my web site on all things real-time and embedded.

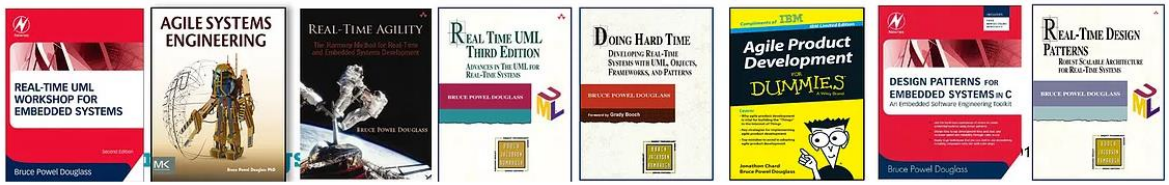
On this site you will find papers, presentations, models, forums for questions / discussions, and links (lots of links) to areas of interest, such as

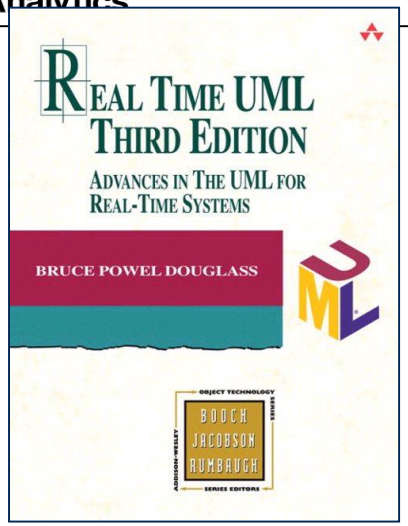
- Developing Embedded Software
- Model-Driven Development for Real-Time Systems
- Model-Based Systems Engineering
- Safety Analysis and Design
- Agile Methods for Embedded Software
- Agile Methods for Systems Engineering
- The Harmony agile Model-Based Systems Engineering process
- The Harmony agile Embedded Software Development process
- Models and profiles I've developed and authored
- List and links to many of my books.

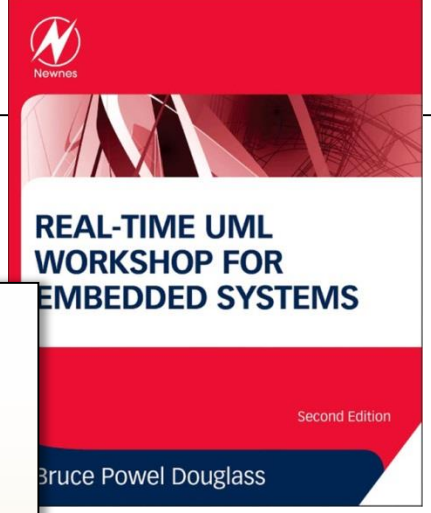
The menu at the top of each page either takes you to the relevant page or to a list of relevant pages.

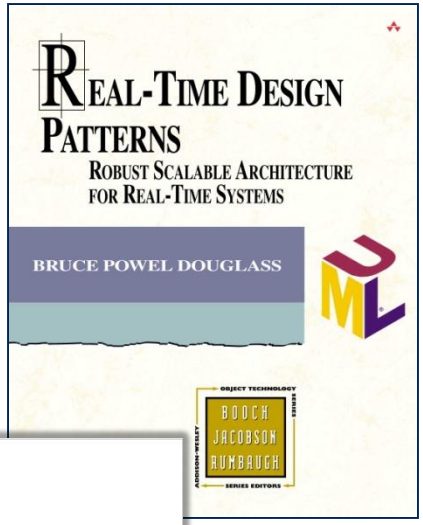
There is even a members only site for those who want to access to even more stuff. I teach and consult on all these topics - see my About page.

Policy on Using These Materials
All materials on this web site are free for reuse and distribution, provided their source (me or this web site) is appropriately attributed. I retain sole copyright.

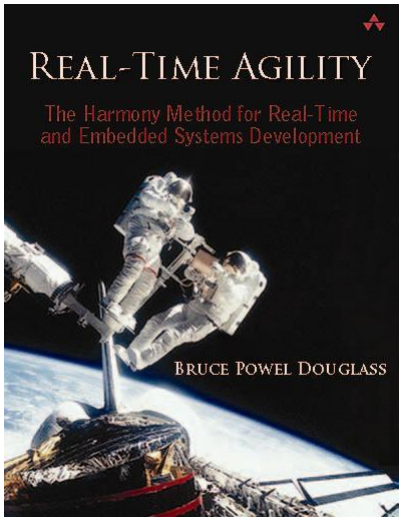


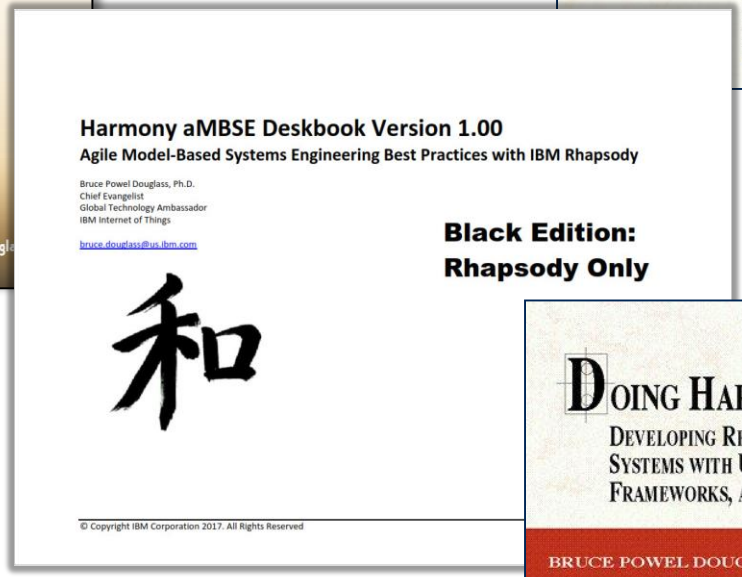

REAL TIME UML
THIRD EDITION
ADVANCES IN THE UML FOR
REAL-TIME SYSTEMS
BRUCE POWEL DOUGLASS
BOOCH
JACOBSON
RUMBAUGH

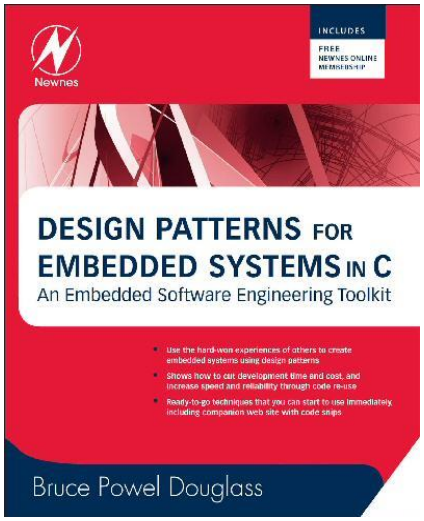

Newnes
REAL-TIME UML
WORKSHOP FOR
EMBEDDED SYSTEMS
Second Edition
Bruce Powel Douglass

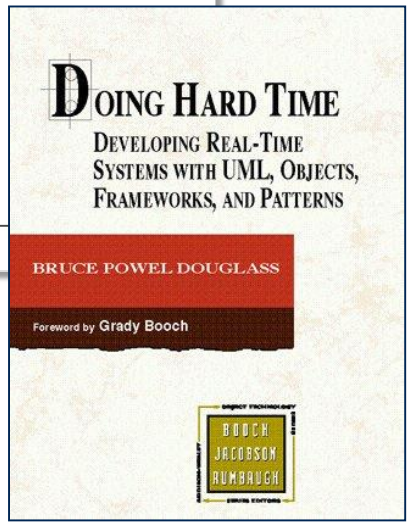

REAL-TIME DESIGN
PATTERNS
ROBUST SCALABLE ARCHITECTURE
FOR REAL-TIME SYSTEMS
BRUCE POWEL DOUGLASS
BOOCH
JACOBSON
RUMBAUGH


AGILE SYSTEMS
ENGINEERING
MK


REAL-TIME AGILITY
The Harmony Method for Real-Time
and Embedded Systems Development
BRUCE POWEL DOUGLASS


Harmony aMBSE Deskbook Version 1.00
Agile Model-Based Systems Engineering Best Practices with IBM Rhapsody
Bruce Powel Douglass, Ph.D.
Chief Evangelist
Global Technology Ambassador
IBM Internet of Things
bruce.douglass@us.ibm.com
Black Edition:
Rhapsody Only
和
© Copyright IBM Corporation 2017. All Rights Reserved


Newnes
INCLUDES
FREE
NEWNES ONLINE
MEMBERSHIP
DESIGN PATTERNS FOR
EMBEDDED SYSTEMS IN C
An Embedded Software Engineering Toolkit
Bruce Powel Douglass


DOING HARD TIME
DEVELOPING REAL-TIME
SYSTEMS WITH UML, OBJECTS,
FRAMEWORKS, AND PATTERNS
BRUCE POWEL DOUGLASS
Foreword by Grady Booch
BOOCH
JACOBSON
RUMBAUGH