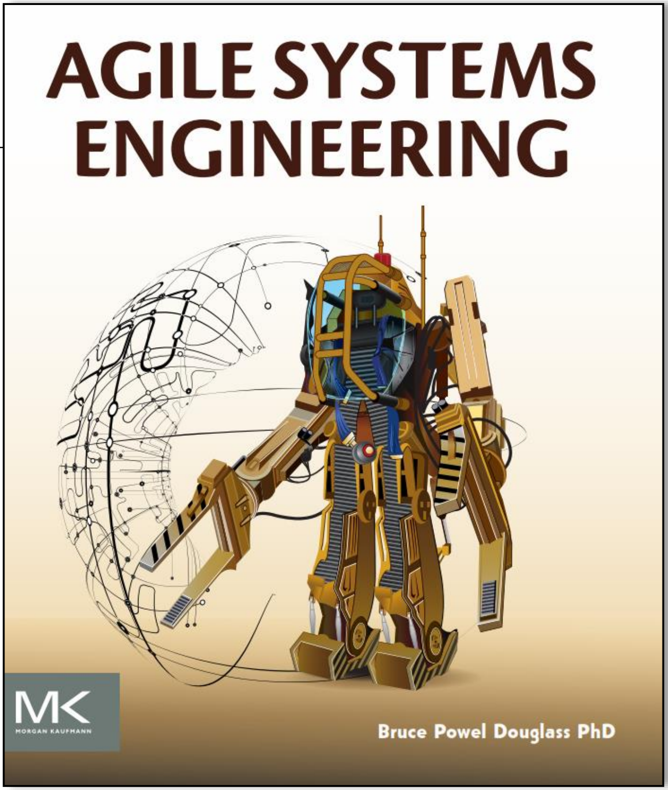
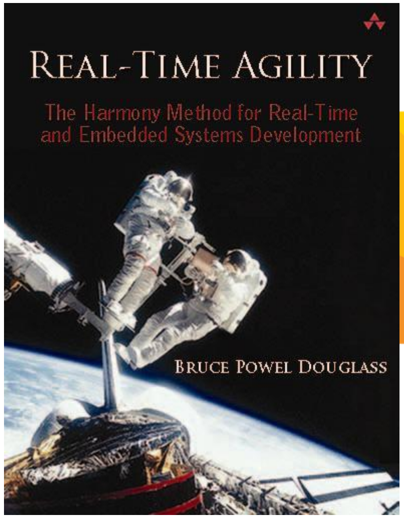
IBM **Analytics**

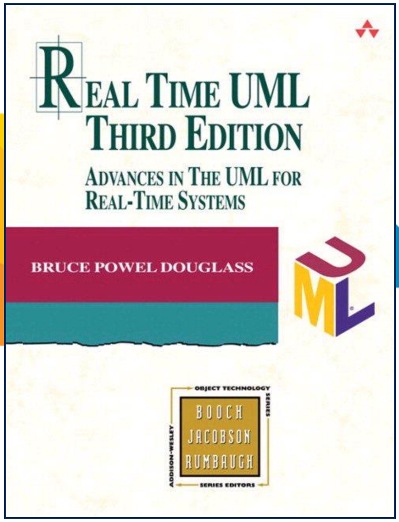
Trade Studies with Rhapsody and SysML

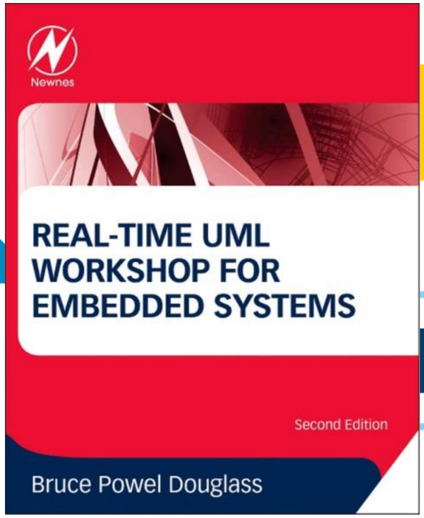
It's just math; how hard can it be?

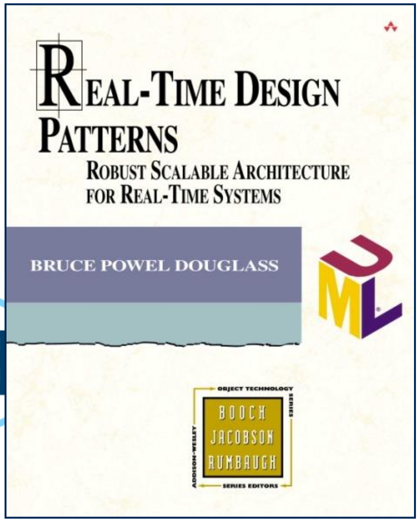
Bruce Powel Douglass, Ph.D.
Chief Evangelist, Global Technology Ambassador
IBM Internet of Things (IoT)
bruce.douglass@us.ibm.com
Twitter: @IronmanBruce
www-01.ibm.com/software/rational/leadership/thought/brucedouglass.html











Harmony aMBSE Overview

Rational. Method Composer

Search this Site:

Team (IBM)

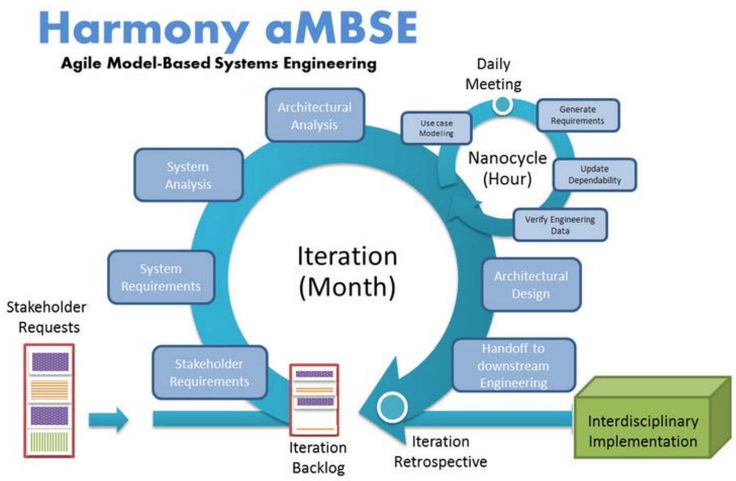
- Welcome to the Rational Harmony Agile Model-Based Systems Engineering
- Getting Started
- Delivery Processes
- Practices
- Roles Sets
- Tasks
- Work Products
- Guidance
- Tools
- Release Info

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.

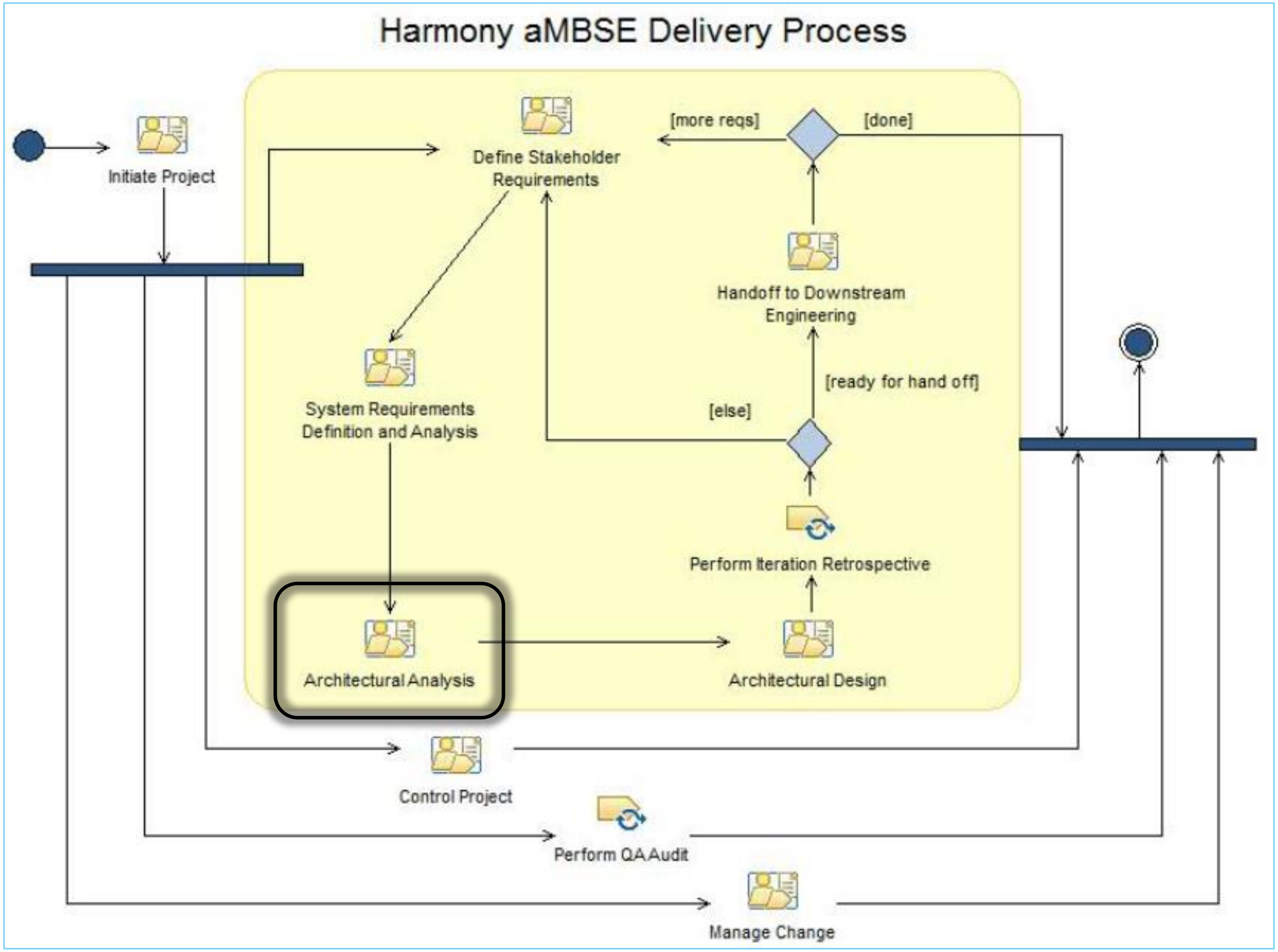
Main Description



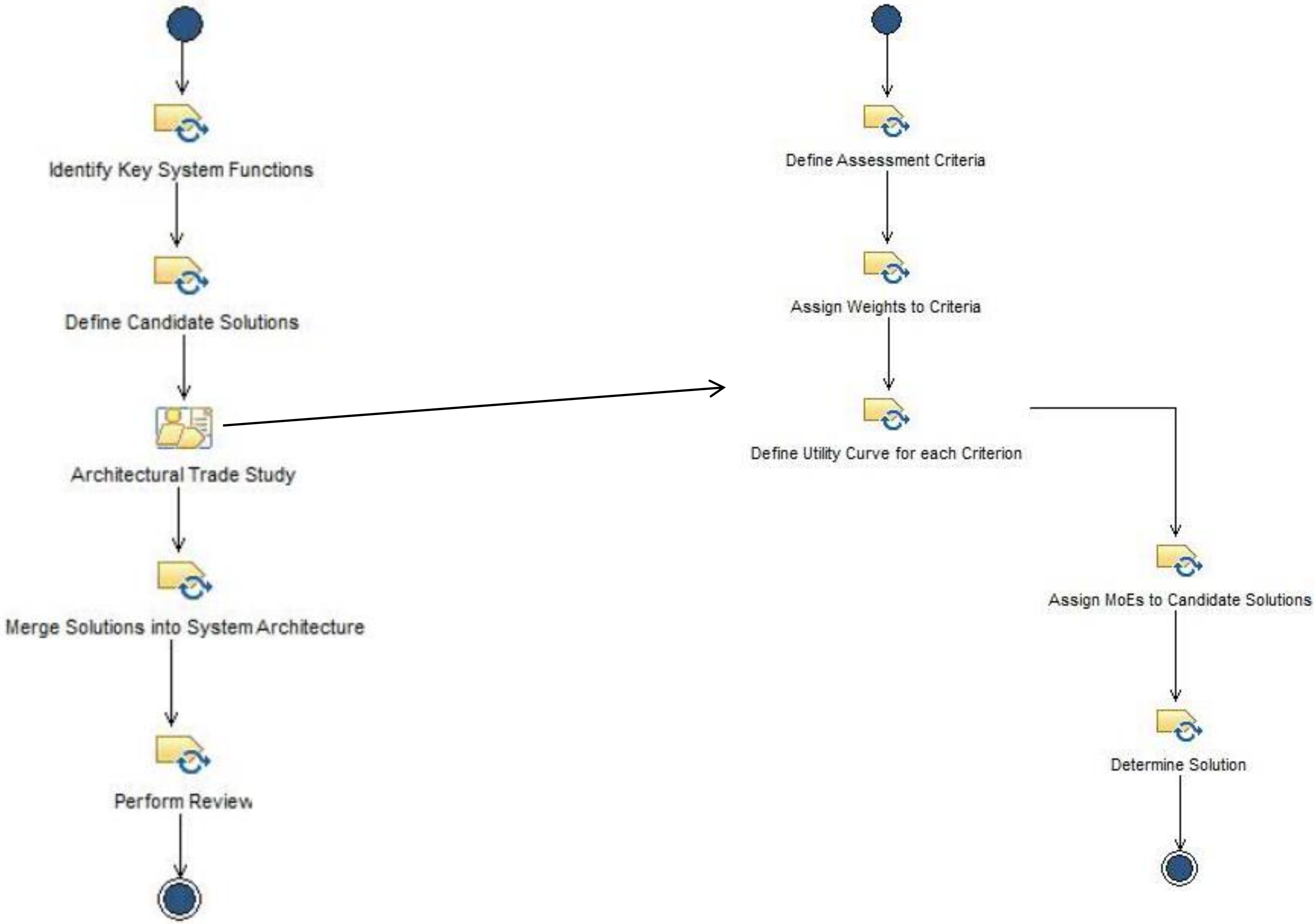
With the initial release of the UML in 1995, systems engineers had a standard language in which they could express requirements, architectures, designs, and other kinds of engineering data. However, there was widespread belief that the Unified Modeling Language (UML) itself was too “software oriented” 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

Harmony aMBSE Process Overview



Systems Architectural Analysis



Identify Key System Functions

- **Key system functions** are system functions that are important, architectural, and subject to optimization.
  - A system function that is important but either not architectural or subject to optimization need not be analyzed for trade offs.
  - To be optimizable, in this case, means the selection of a fundamentally different architectural structure or different technology.
  - For example, if you want to provide motive force for a robot arm, should you use
    - pneumatics,
    - hydraulics, or an
    - electrical motor?
    - All have pros and cons, and a trade study can select which is best for the given system given its requirements and usage context.

Approach 1: Architectural Trade Study (Lightweight)

- Architecture is
  - The sum of the large scale organizational and technological decisions that optimize the system design criteria
  - A trade study examines architectural alternatives to select which is "best"
  - Basic flow:
    - Identify the key system functions
    - Define the design criteria (aka "Measures of Effectiveness" (MoE))
      - Identify the design optimization criteria
      - Rank the criteria in order of importance or criticality
    - Identify the set of candidate solutions
    - Assess each solution a score* against each criteria
      - Compute a score as

$$CandidateScore = \sum C_j W_j$$

      - Where

        C_j is the criticality (MoE) of design criteria j

        W_j is the degree to which the criteria is optimized by the candidate solution

* This score is essentially an estimation of where the solution fits on the utilization curve. More on this later.

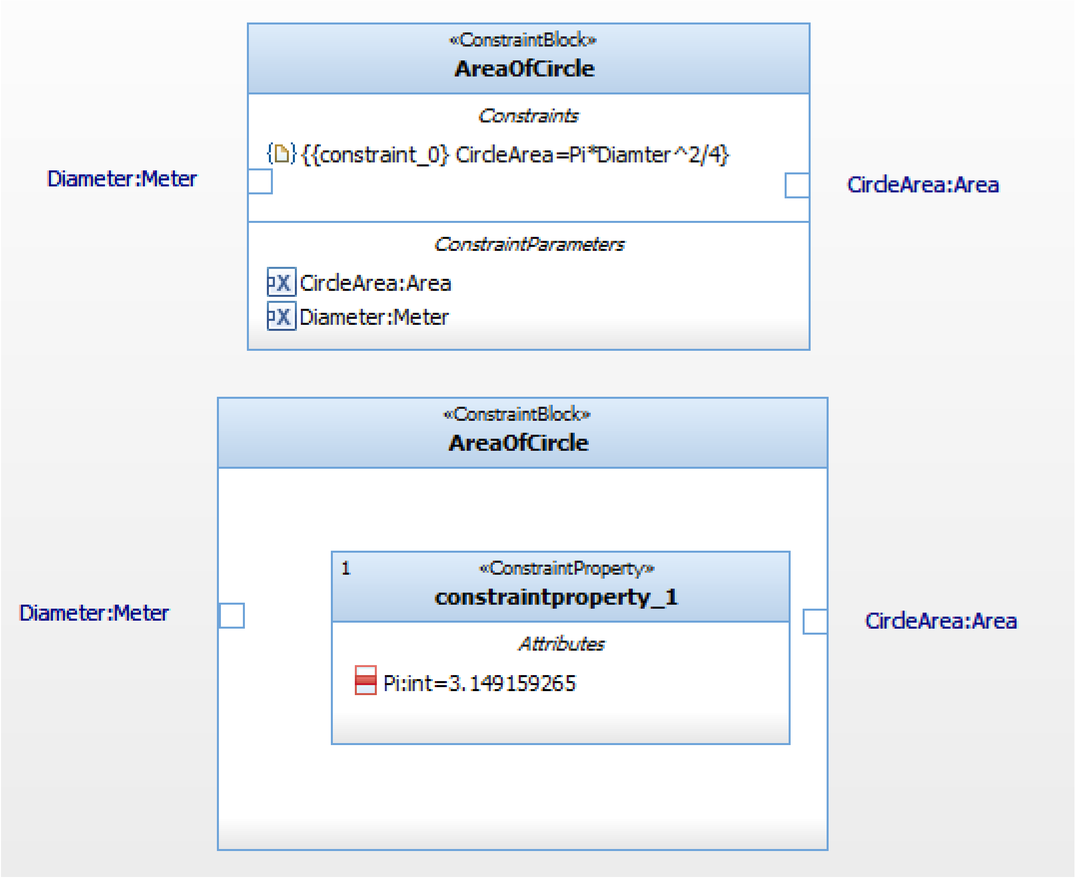
Architectural Trade Study Example

Candidate Solutions	Solution Criteria										Weighted Total
	Power Consumption (W_1= 0.3)		Recurring Cost (W_2 = 0.2)		Robustness (W_3 = 0.15)		Development Cost (W_4 = 0.1		Security W_5 = 0.25)		
	MoE	Score	MoE	Score	MoE	Score	MoE	Score	MoE	Score	
Gigabit Ethernet Bus	2	0.6	2.7	0.54	4	0.6	8	0.8	4	1.0	3.54
1553 Bus	3	0.9	4	0.8	10	1.5	1.5	0.15	6	1.5	4.85
CAN Bus	6	1.8	8	1.6	7	1.05	3	.3	1	0.25	5.0

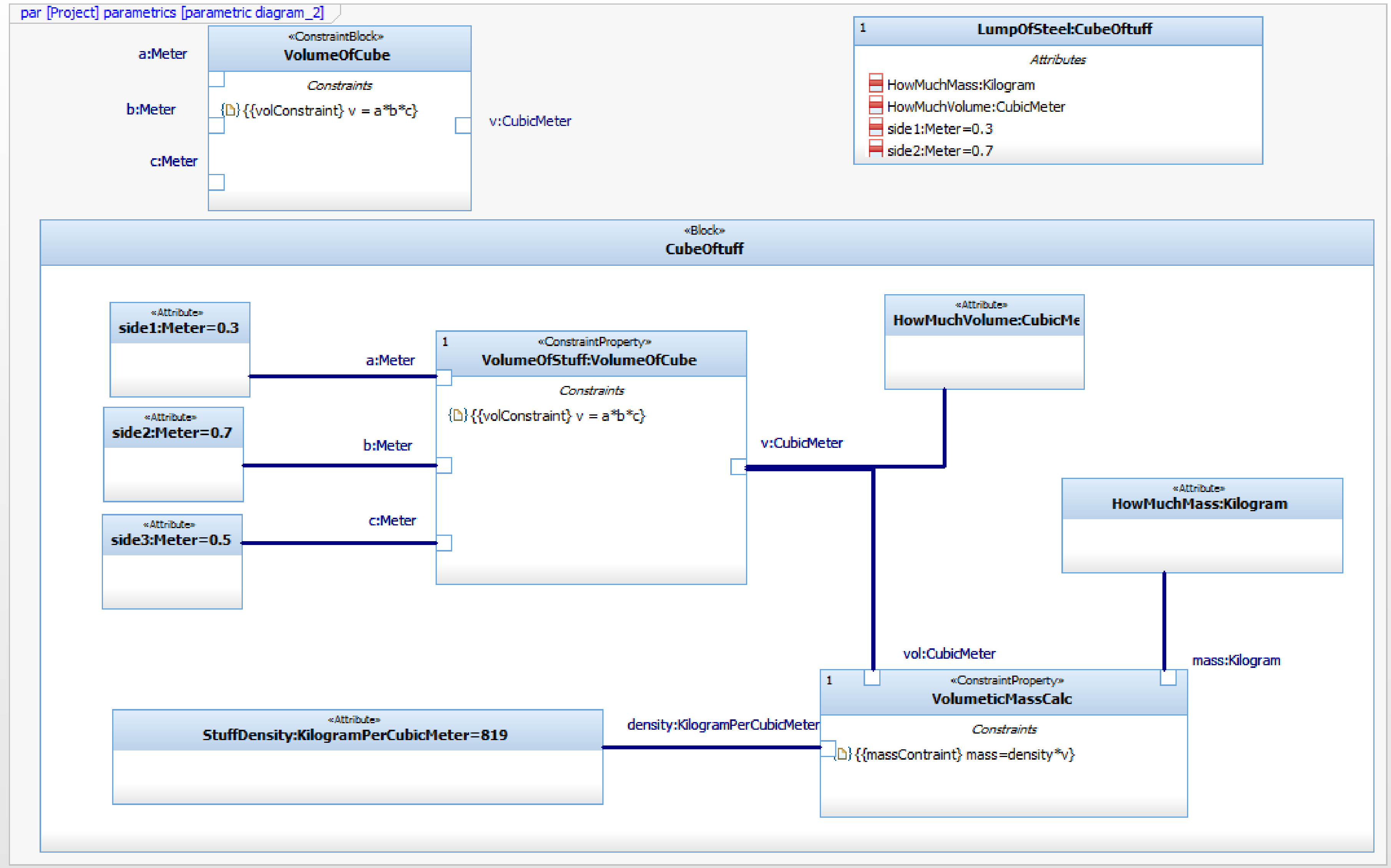
Estimate the utility function to determine MoE scores (degree of optimization of a particular solution)

Let’s get rigorous! Parametric Diagrams FTW

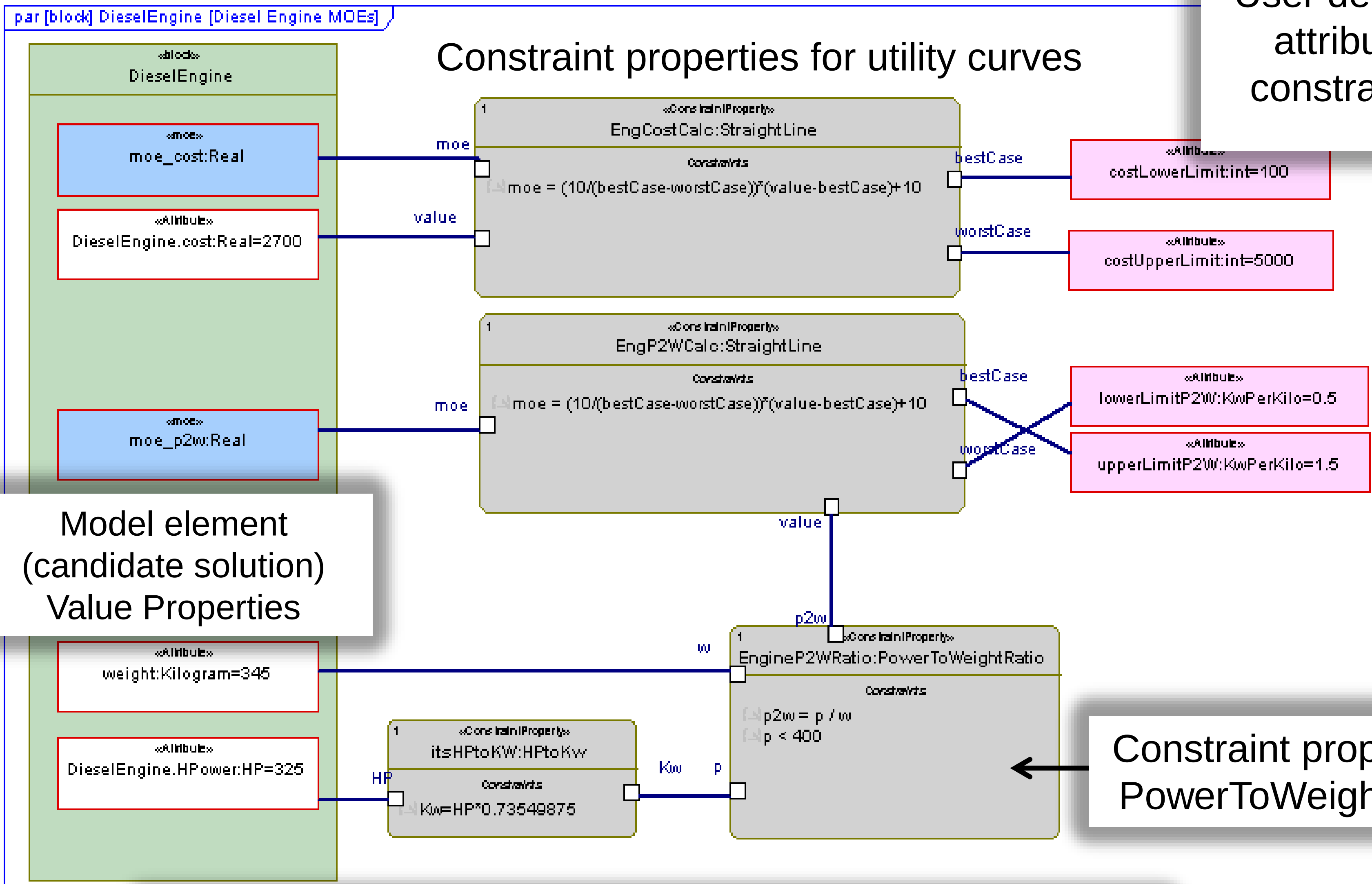
- Relates
  - Constraints (a limitation)
  - Constraint Blocks (constraints with input & output parameters)
  - Constraint property (usage or instance of Constraint block)
  - Variables or attributes
- Used for
  - Computation
  - Evaluation of trade offs



Parametric Diagrams



Trade study example



User defined attribute constraints

Model element (candidate solution) Value Properties

Constraint property of PowerToWeightRatio

White are inputs, pink are ranges, blue are outputs

Approach 2: Using parametric diagrams with mathematical analysis

- When used with a mathematical modeling engine, parametrics become very powerful.
  - Used for:
    - Architecture Optimization
    - Trade Studies
    - Engineering calculations (including unit conversion)
- Typical mathematical modeling tools are:-
    - Modelica (supported by PCE profile)
    - Maxima (supported by PCE profile)
    - Simulink (supporte by the Simulink Profile)
    - Excel (limited, but supported by Harmony SE-Toolkit)
- These tools can be linked to the parametric model and evaluate the constraints.

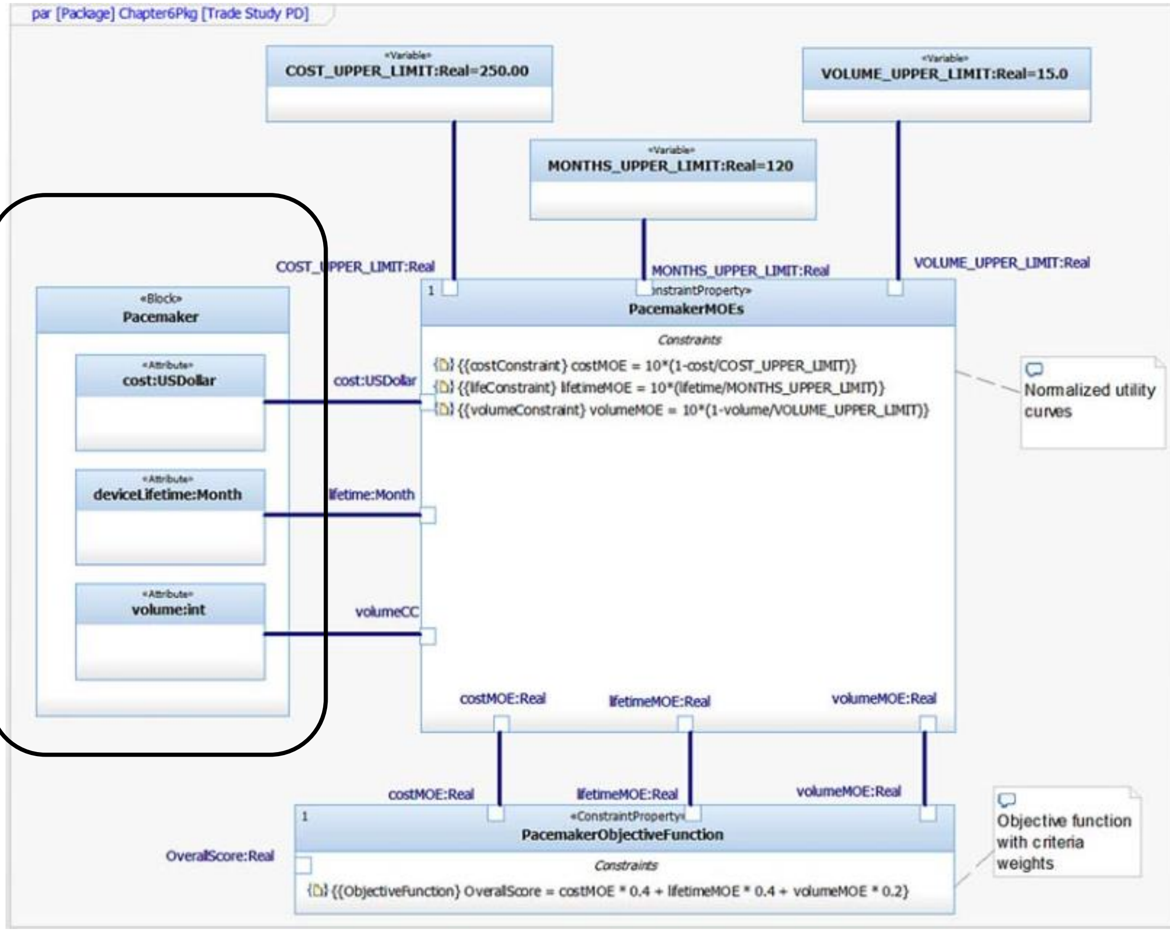
The role of Instance Specifications in Evaluation of Parametrics

- In parametric diagrams, blocks can be shown that have attributes (value properties)
- To do evaluation, *instance specifications* can assign values to these attributes for the purpose of evaluation of different cases

This Pacemaker block has attributes

- cost
- deviceLifetime
- volume

This block is special-purpose, created for this analysis



The role of Instance Specifications in Evaluation of Parametrics

- We can create **instance specifications** to give these attributes (instance slots) actual values

- And then evalute the parametric functions on this basis:

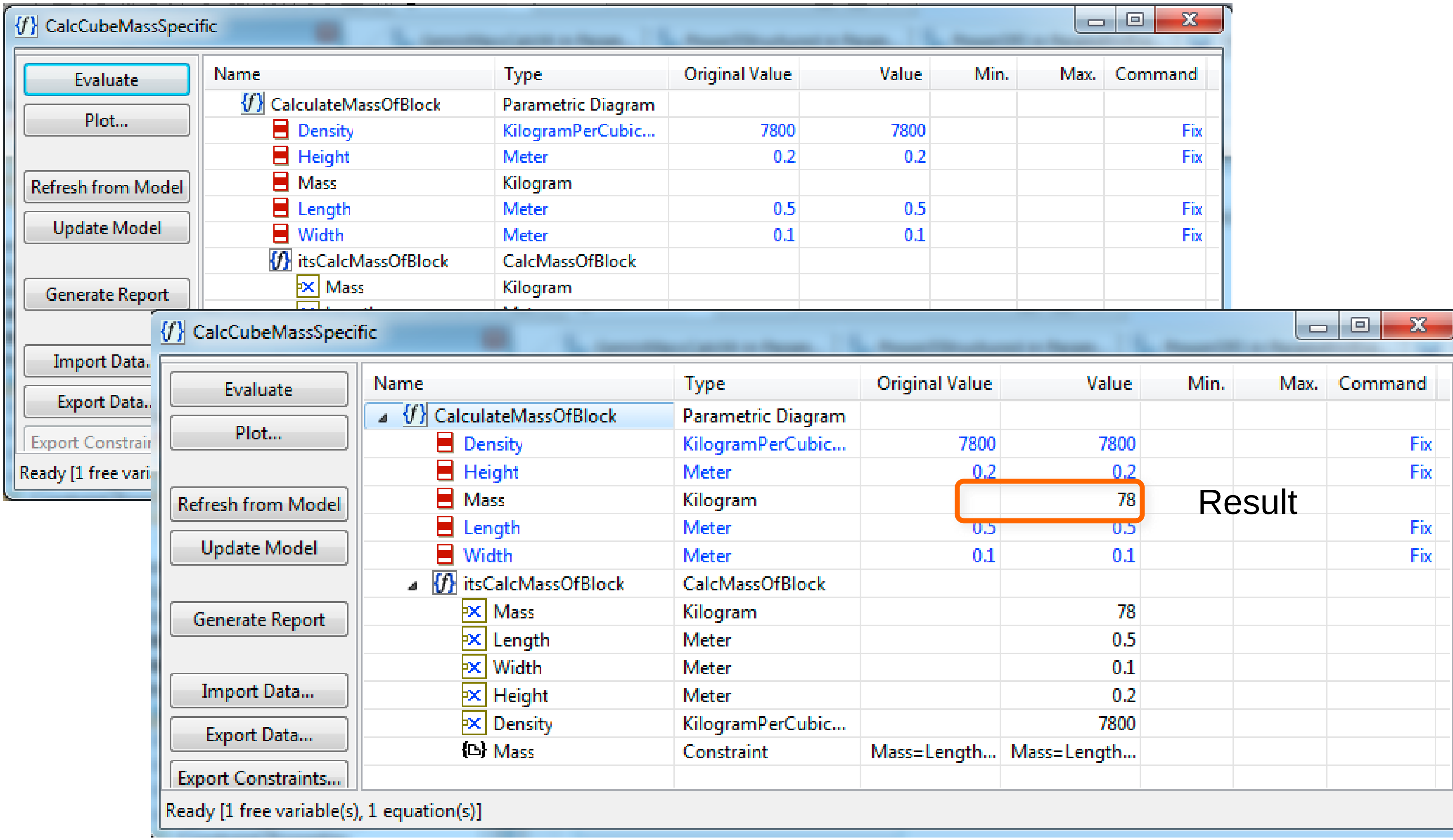
volumeConstraint	Constraint	volumeMOE ...	volumeMOE ...
PacemakerObjectiveFunc	PacemakerObjectiv...		
costMOE	Real		0
lifetimeMOE	Real		10
volumeMOE	Real		0
OverallScore	Real		4
ObjectiveFunction	Constraint	OverallScore ...	OverallScore ...

volumeConstraint	Constraint	volumeMOE ...	volumeMOE ...
PacemakerObjectiveFunc	PacemakerObjectiv...		
costMOE	Real		5.6000000000...
lifetimeMOE	Real		6.6666666666...
volumeMOE	Real		6
OverallScore	Real		6.1066666666...
ObjectiveFunction	Constraint	OverallScore ...	OverallScore ...

volumeConstraint	Constraint	volumeMOE ...	volumeMOE ...
PacemakerObjectiveFunc	PacemakerObjectiv...		
costMOE	Real		4
lifetimeMOE	Real		8.3333333333...
volumeMOE	Real		3.4666666666...
OverallScore	Real		5.6266666666...
ObjectiveFunction	Constraint	OverallScore ...	OverallScore ...

Doing mathematical analysis

- Invoke one of the mathematical modeling tools and do the analysis
- Push result back into the model or export



Utility Curves: Calculating the Measure of Effectiveness (MoE)

The utility curve computes the "goodness" score of a solution.

Commonly,

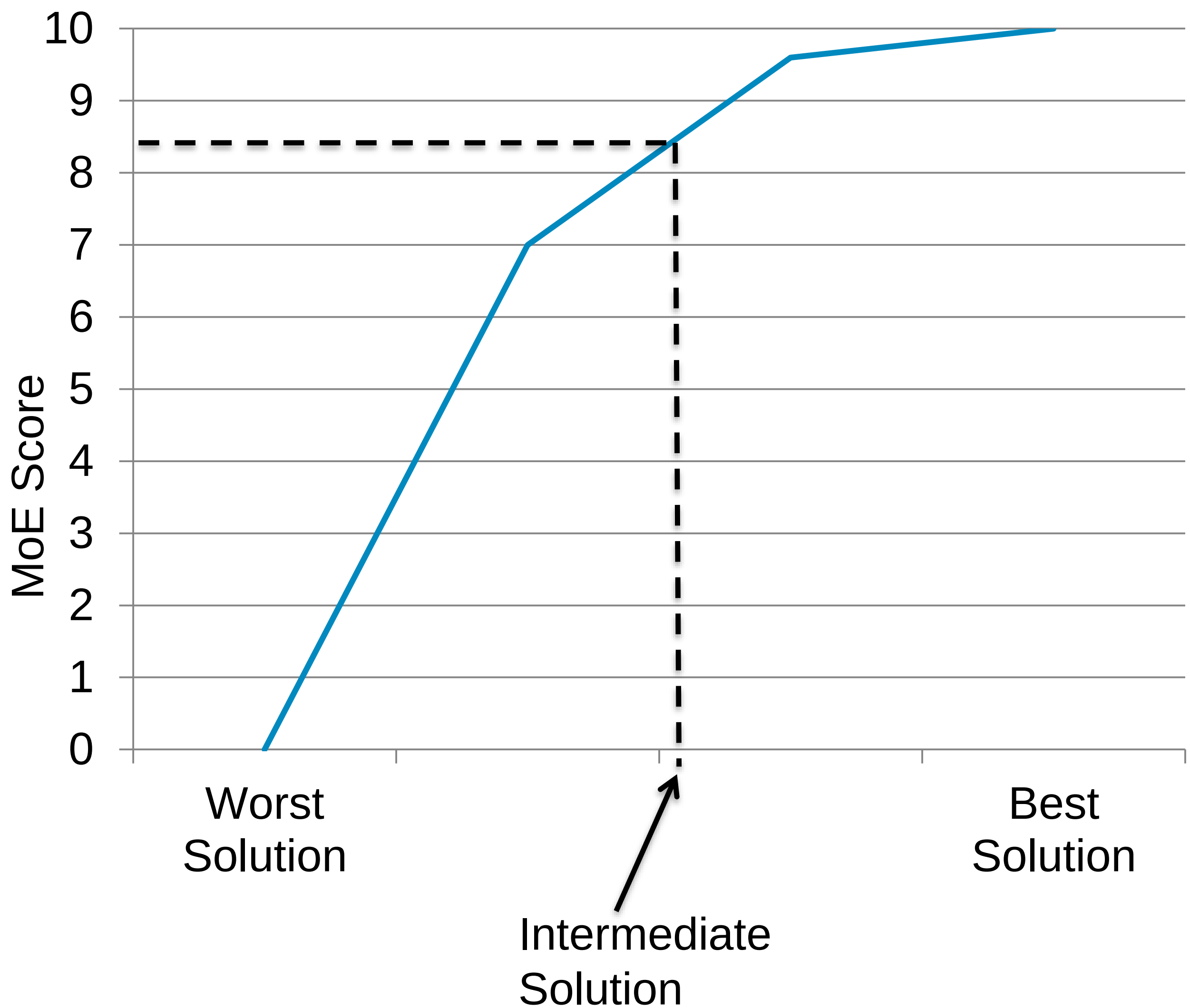
- Computed output scores normalized to the range of
  - 0 (for worst case)
  - 10 (for best case)

- Most commonly, linear utility functions are used in which lines are drawn using these as the end points

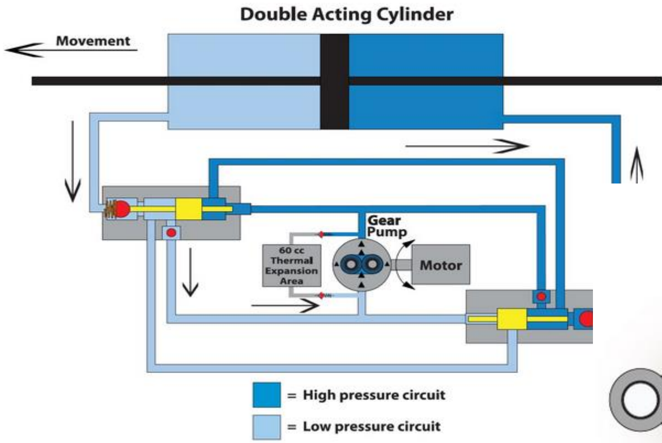
- For solutions with intermediate value, their MoE Score can be determined from the function

- For example, a utility function for Energy Production might be
  - 5hp (worst case) → 0
  - 150 hp (best case) → 10
  - A solution that produces 100hp might have a computed score of 8.25

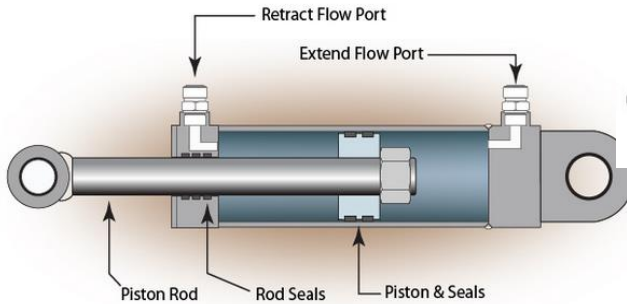
- Note: You must define how the input values will be determined:
  - Measured in the lab?
  - Estimated?
  - Use manufacturer's data?



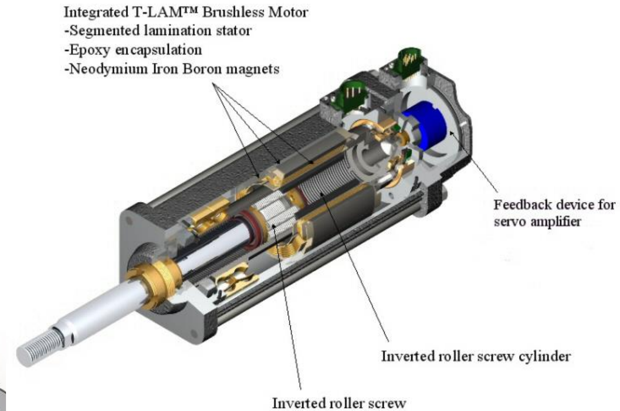
T-Wrecks: Determine Candidate Solutions



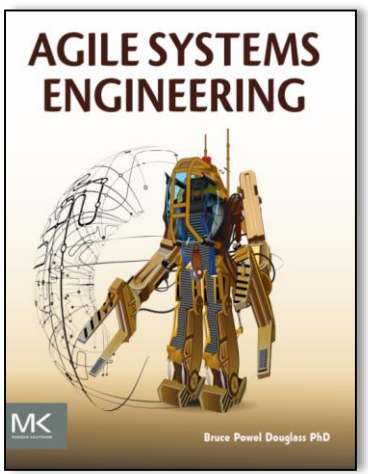
Electrohydraulic actuator



Linear Pneumatic actuator



Direct current motor



T-Wrecks assessment criteria weights			
Function/Feature	Criterion	Means of Determination	Weight
Proximity Scanning	Cost	Use manufacturer's data	0.10
	Weight	Use manufacturer's data	0.20
	Near range accuracy	Measured in lab from samples	0.30
	Angular accuracy	Taken from manufacturer's spec	0.30
	Response time	Calculation based on manufacturer's data	0.10
Joint movement	Cost	Use manufacturer's data	0.1
	Weight	Use manufacturer's data	0.05
	Response time	Use manufacturer's data	0.2
	Durability	Use manufacturer's data	0.65

Computing Measures of Effectiveness (MOEs)

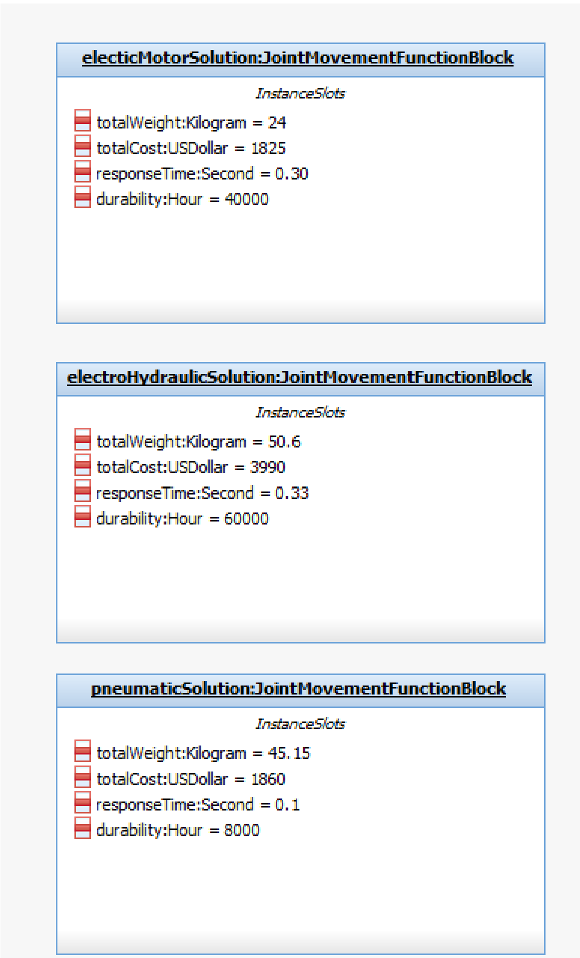
T-Wrecks assessment criteria weights		
Function/Feature	**Criterion**	**Utility curve**
Proximity Scanning	Cost	psCostMOE = -1.67×10^{-4} * psTotalCost + 10.01
	Weight	psWeightMOE = -1.10 psTotalWeight + 14.40
	Near range accuracy	NRAccuracyMOE = -10 * accuracy + 20
	Angular accuracy	angAccuracyMOE= -1.121 * AngAccuracy + 10.09
	Response time	psResponseTimeMOE = 10-exp(20*responseTime)/110
Joint Movement	Cost	CostMOE = -0.0462 * jmTotalCost + 18.43
	Weight	WeightMOE = -jmTotalWeight * 0.3759 + 22.389
	Response Time	rtResponseTimeMOE = 10-exp(20*responseTime)/110
	Durability	DurabilityMOE = MTBF/5200 - 1.538

What was done in this case was the best solution was given an MOE value of 10 and the worst was given a value of 0. Then a line was computed passing through the two points. The MOE equation is the equation for that line.

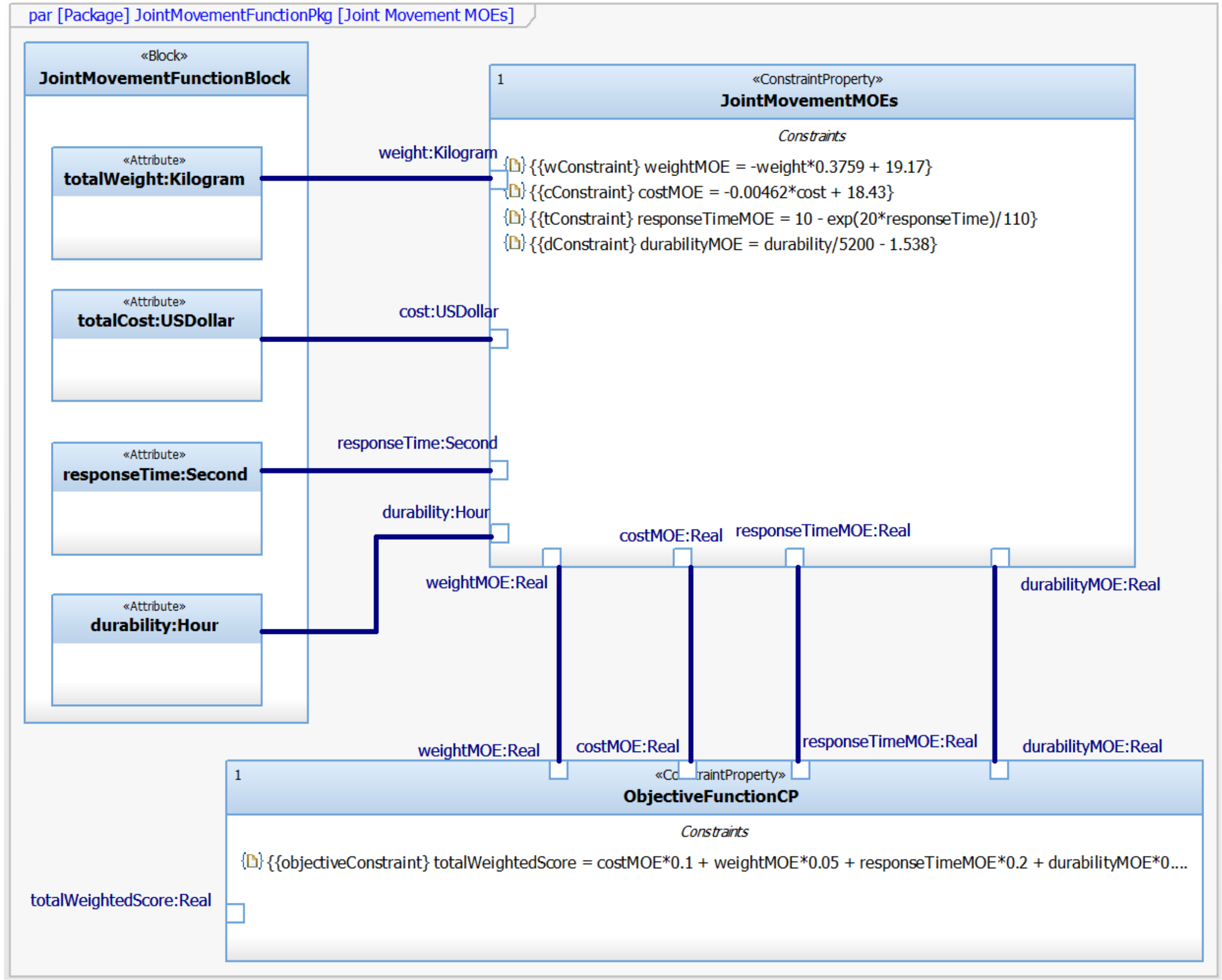
In the case of Response Time, an exponential curve was used instead.

Performing the trade study with a Parametric Diagram

Instance Specs of Solutions

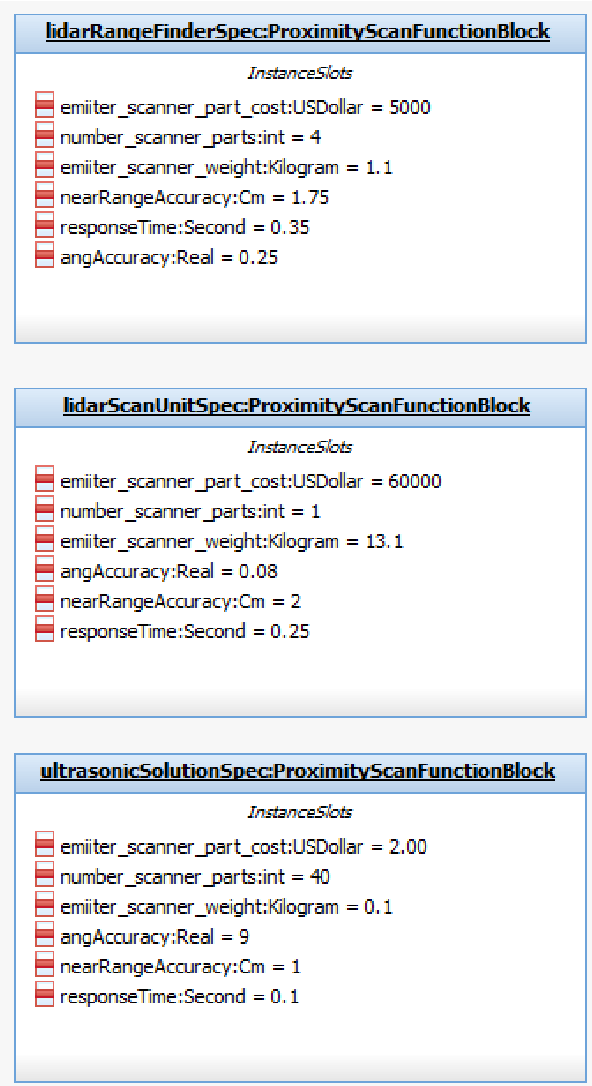


Parametric Expression of Trade Analysis

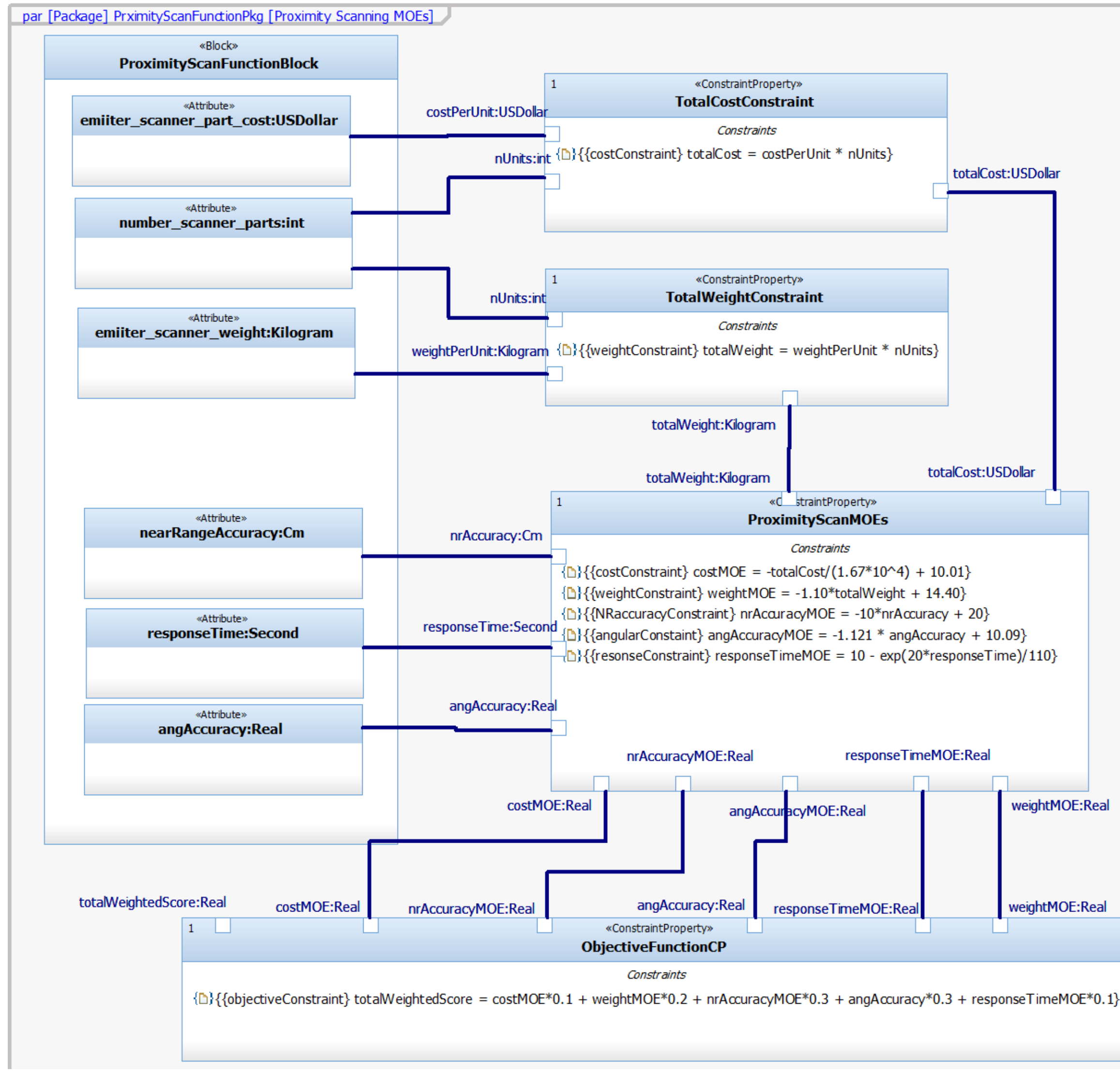


Performing the trade study with a Parametric Diagram

Instance Specs of Solutions



Parametric Expression of Trade Analysis



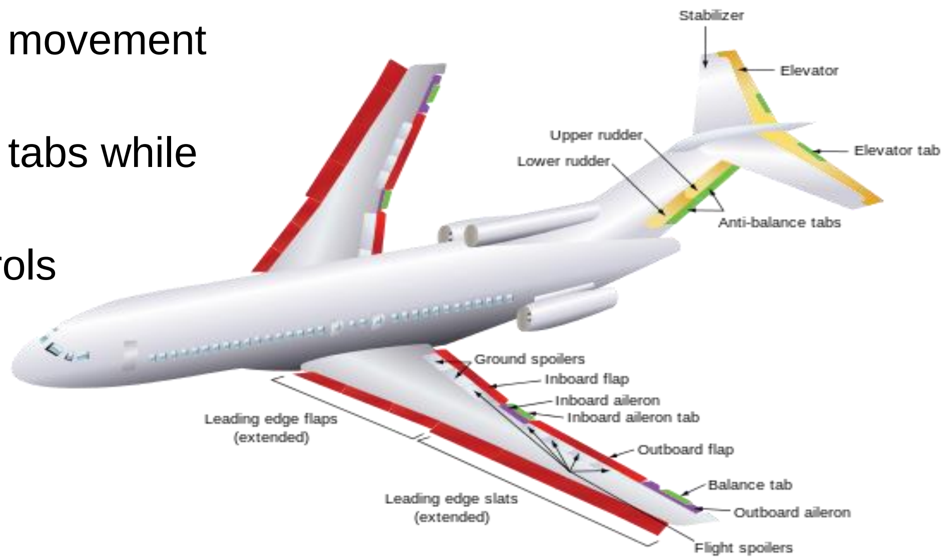
Performing the trade study

T-Wrecks Computed Total Weighted Scores		
Function/Feature	Solution	Total Weighted Score
Proximity Scanning	**Solution 1 (Ultrasonic)**	**6.994**
	Solution 2 (lidar scan unit)	4.505
	Solution 3 (lidar range finder)	6.489
Joint Movement	Solution 1 (electric motors)	6.77
	Solution 2 (electro-hydraulic)	**7.17**
	Solution 3 (pneumatic)	3.08

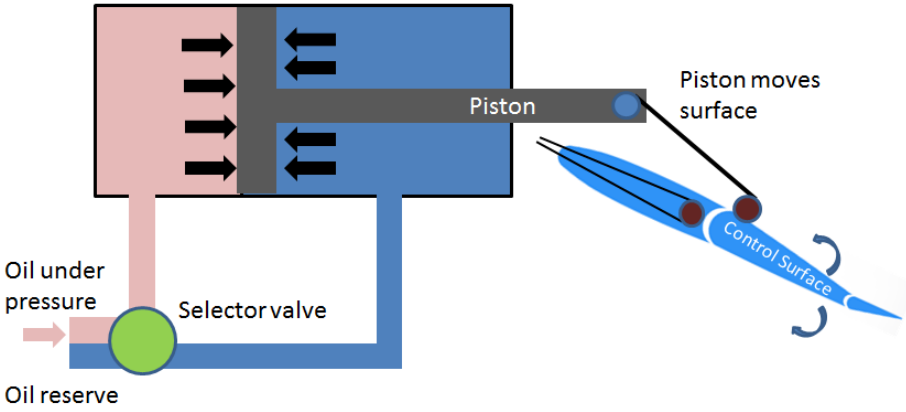
These results were computed from Rhapsody with the PCE profile added, linking in Maxima® to perform the computations and determine the total weighted scores

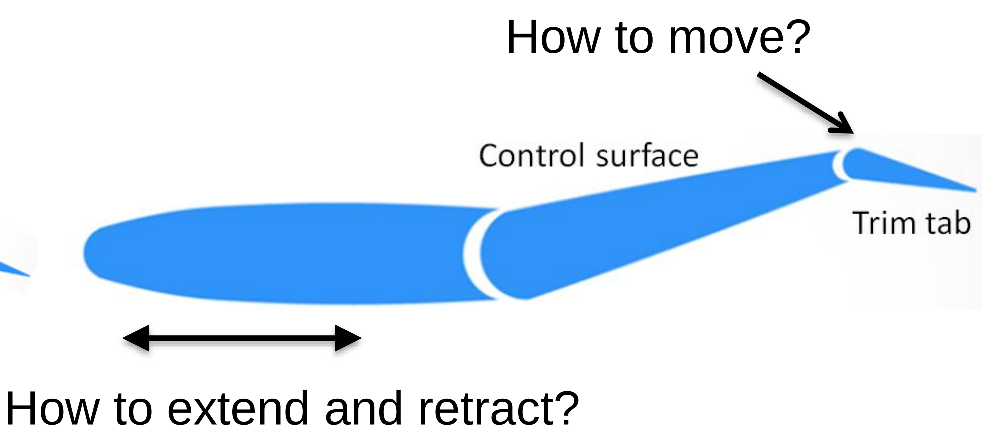
Approach 3: Using the Harmony SE Toolkit

- This system uses hydraulics for the primary movement of the 36 control surfaces
- However, many of these surfaces have trim tabs while others may also extend and retract
- How should these secondary surfaces controls be accomplished?

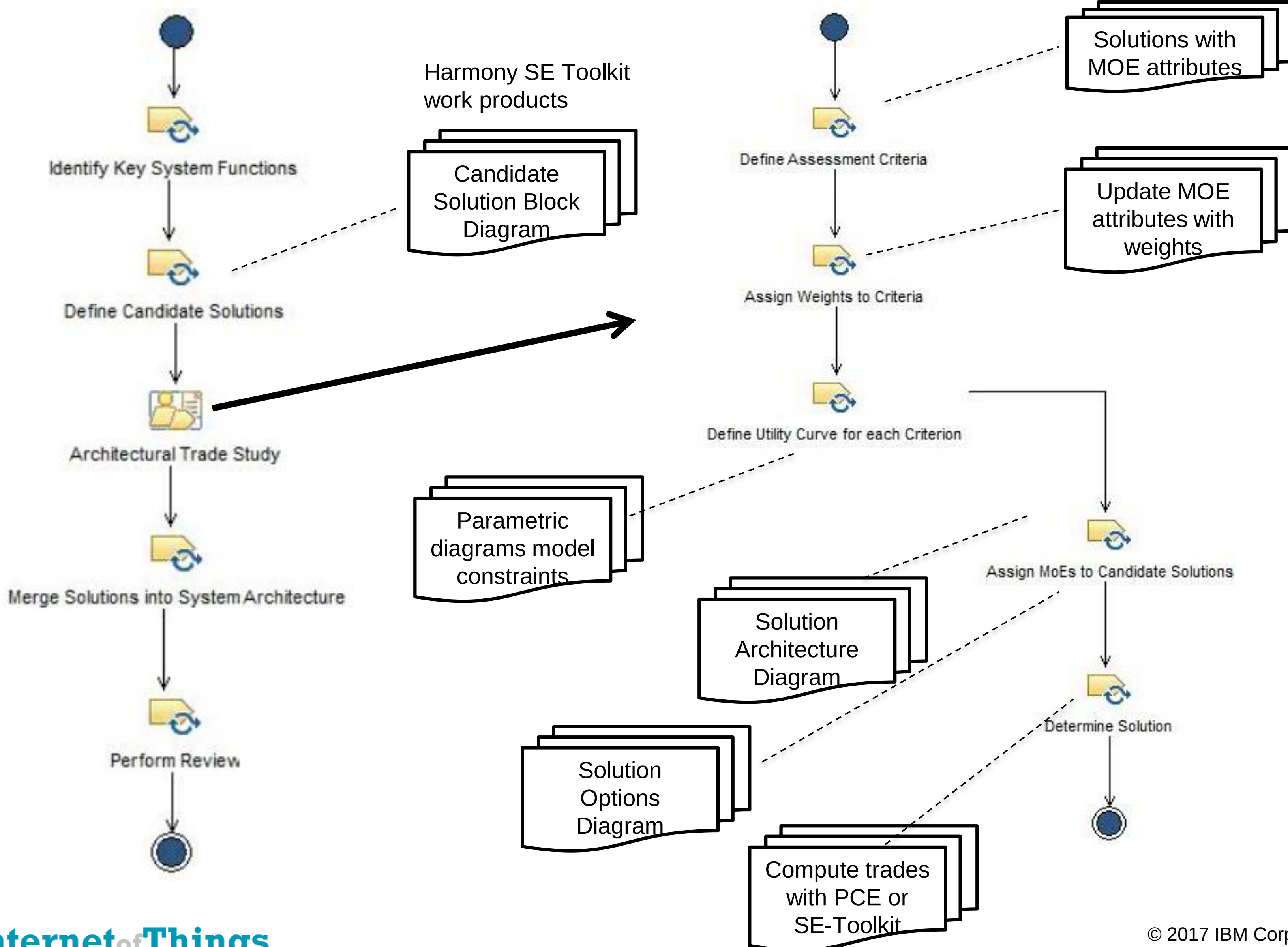


Primary movement via hydraulics





Systems Architectural Analysis with Harmony SE Toolkit

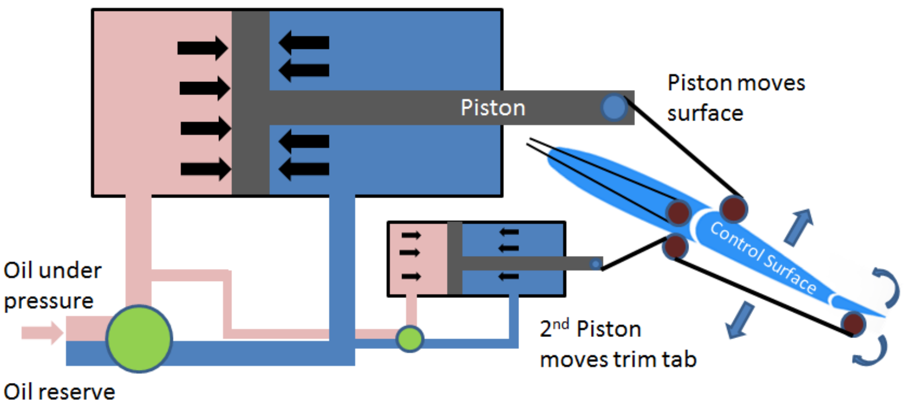


Answer: Identify Key System Functions

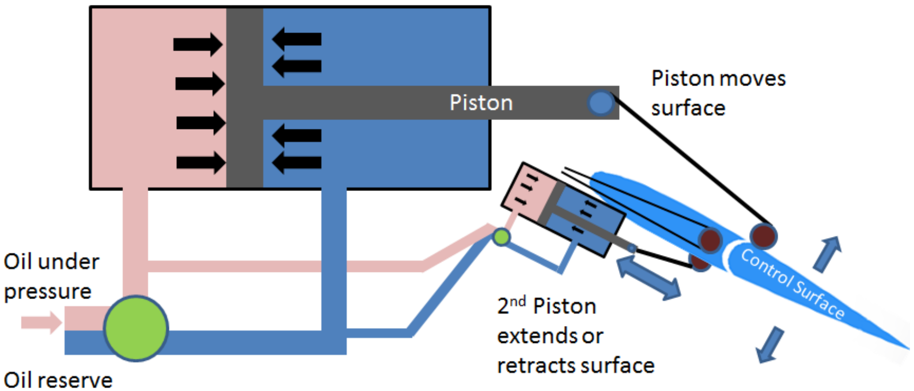
- Measurement of surface movement position
- Measurement of surface movement timing
- Error date storage
- Checking power status
- Checking hydraulic status
- Checking software integrity
- Communicating with the aircraft AMS, Power, and Hydraulic systems (presumably they have an already defined interface).
- **Control of surface movement**
  - **Specifically trim tab movement and extension/retraction**

Answer: Define Candidate Solutions

- Solution 1: Hydraulic: extend existing aircraft hydraulic system



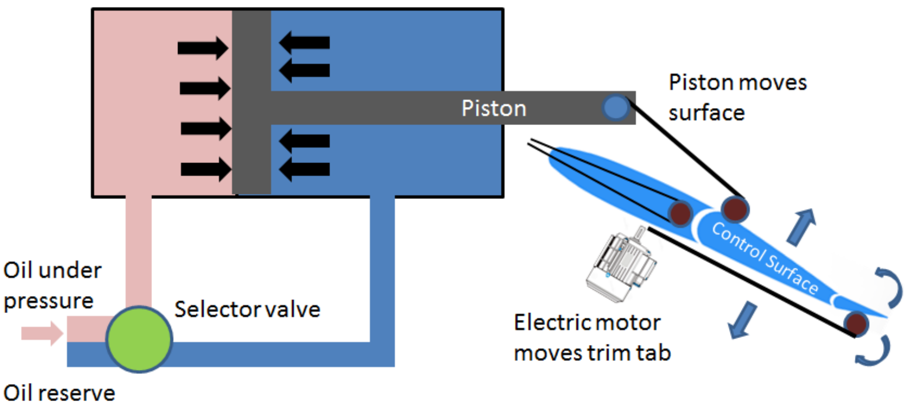
Trim tab movement



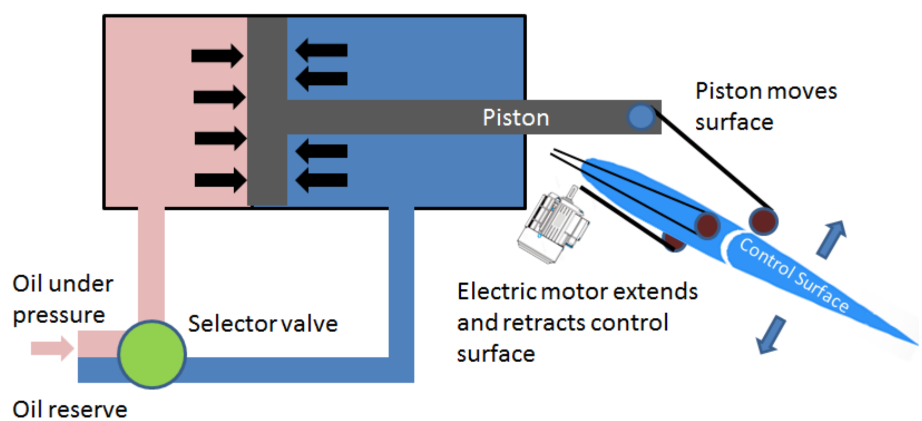
Surface extension/retraction

Answer: Define Candidate Solutions

- Solution 2: Electric Motor: Add electic motors for trim tab and extension movement



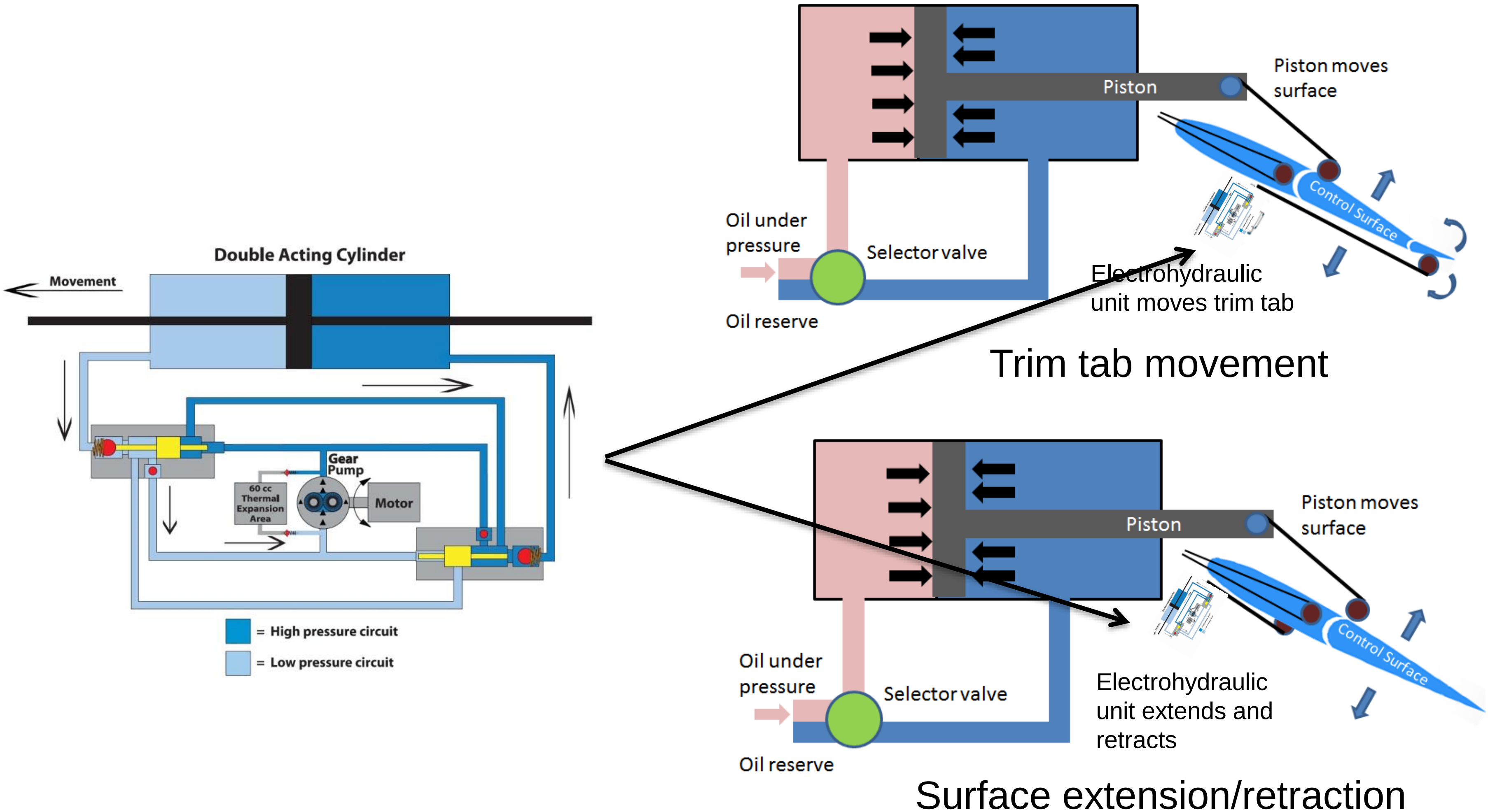
Trim tab movement



Surface extension/retraction

Answer: Define Candidate Solutions

- Solution 3: Add Self-Contained Electro Hydraulic Actuator

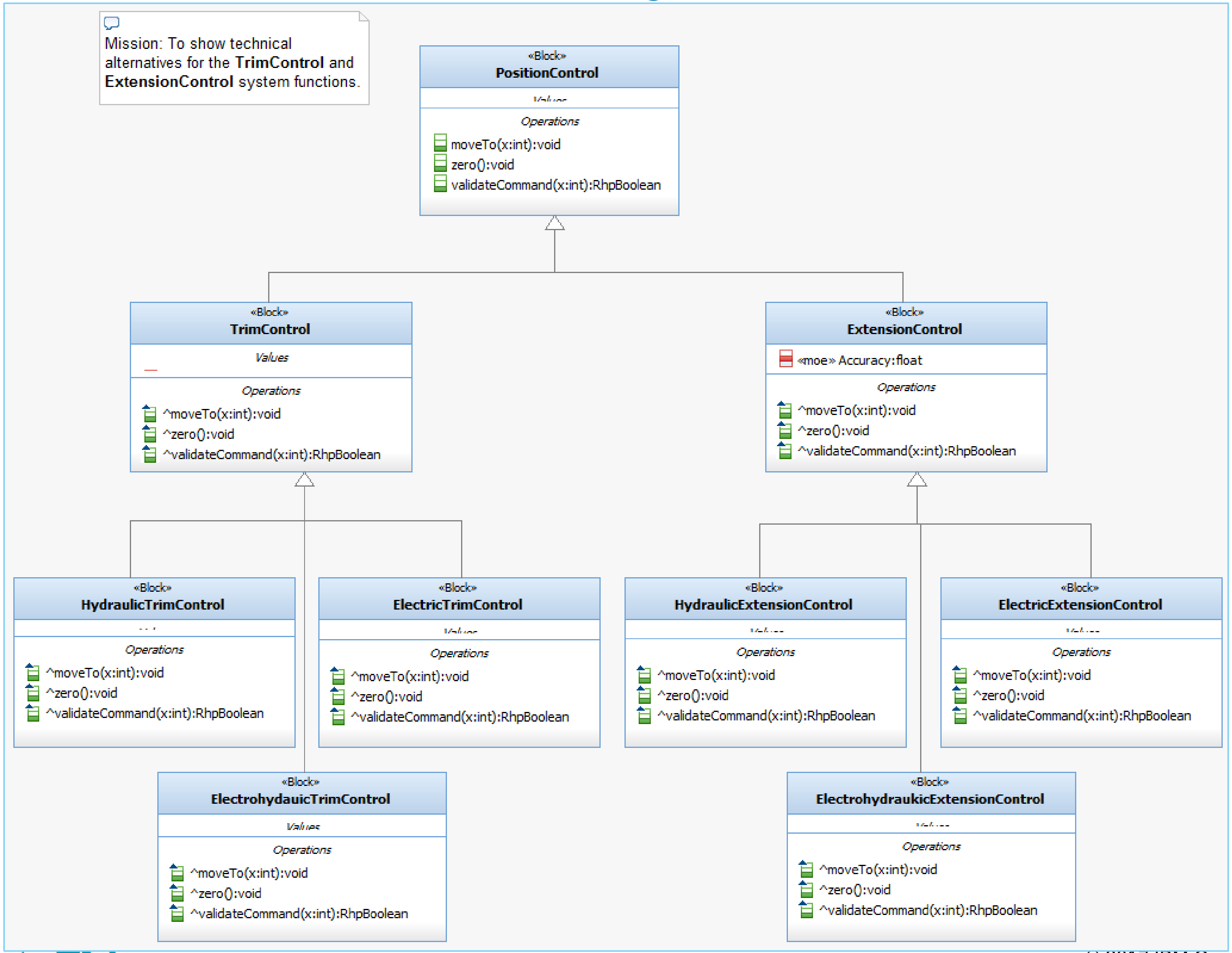


Define Candidate Solutions

- Create a new package; **TrimControlTradeStudy**.
- In the new package, add a new block definition diagram named **Trim Control Alternatives**.
- On this diagram, add new blocks:
  - **PositionControl**
  - **TrimControl**
  - **HydraulicTrimControl**
  - **ElectricTrimControl**
  - **ElectriHydraulicTrimControl**
  - **Extensioncontrol**
  - **HydraulicExtensionControl**
  - **ElectricExtensioncontrol**
  - **ElectroHydraulicExtensionControl**
- The **PositionControl** block has two operations that are aspects of this: Add
  - **moveTo(x: int)**
  - **zero()**
  - **ValidateCommand(x: int)**
- Add the generalization relations (e.g. **TrimControl** is a type of **PositionControl**)

Define Candidate Solutions Block Diagram

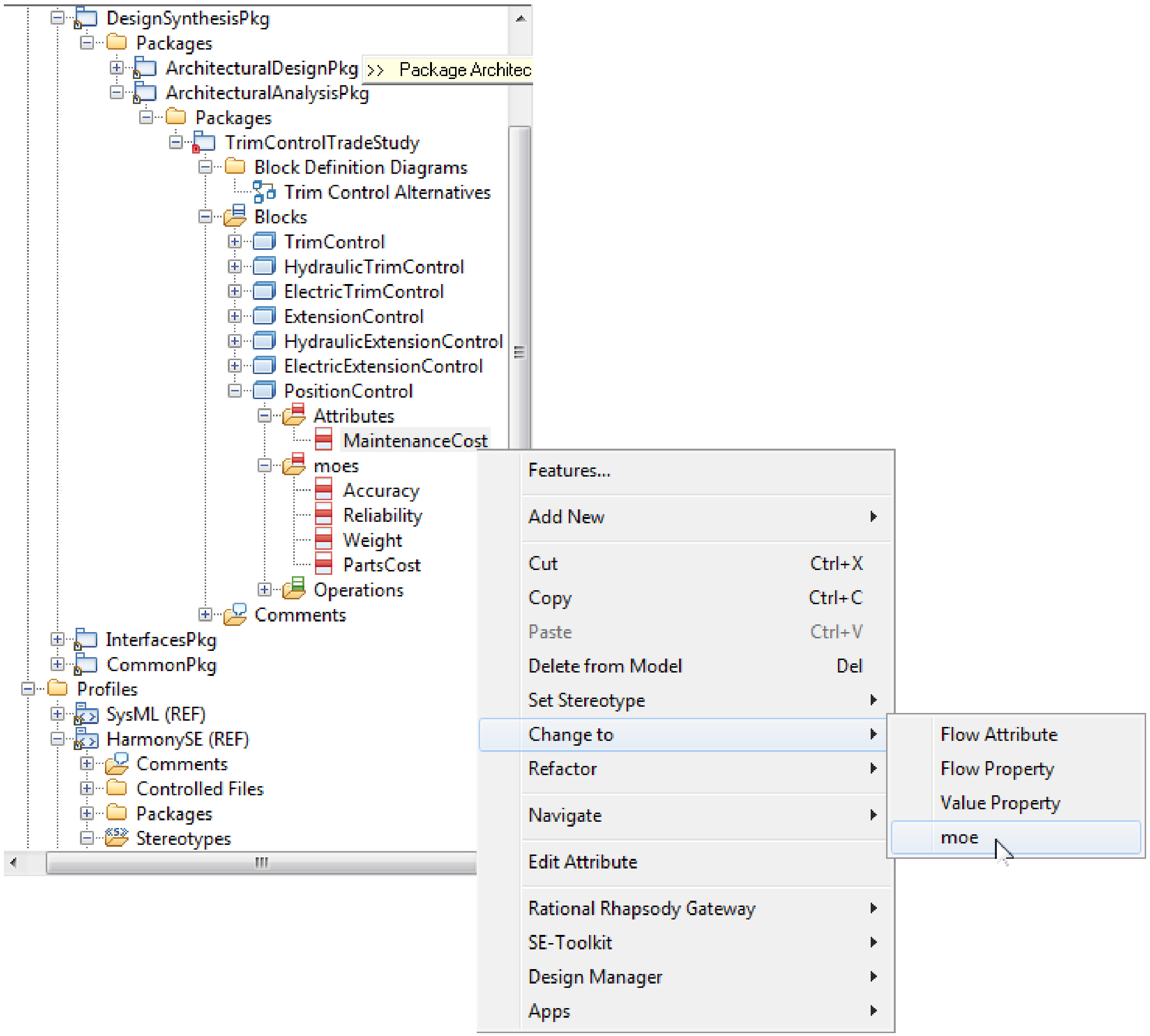
Candidate solutions

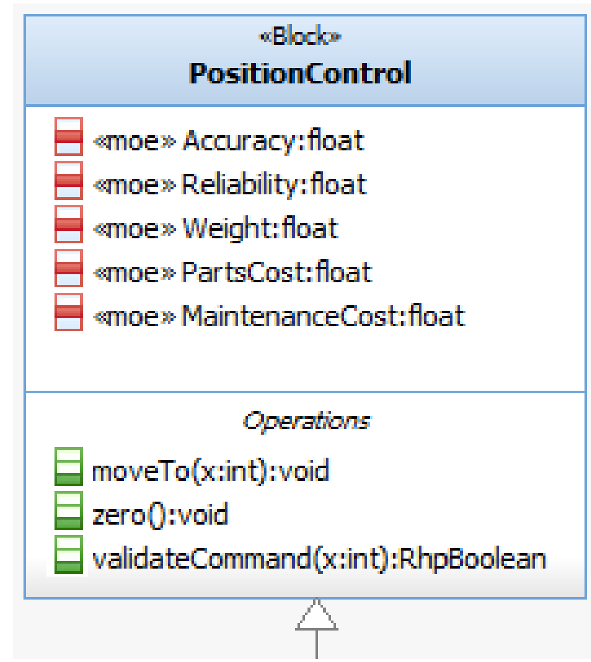


Define Assessment Criteria

- The key to selecting one technical solution over another is the identification of the assessment criteria. Good assessment criteria allow us to distinguish between good and better solutions in how they effect important, measureable properties of the system. In our case, there are five assessment criteria:
  - Accuracy
  - Weight
  - Reliability
  - Parts Cost
  - Maintenance Cost
- Add these to the **PositionControl** block as attributes (of type *float* or *double*), and
  - Then *in the browser*, select each attribute and *Change To* an **moe** (supplied stereotype)
  - **moe** is a new metaclass defined in the HarmonySE profile. It brings along a *tag* named **weight**.

Define Assessment Criteria





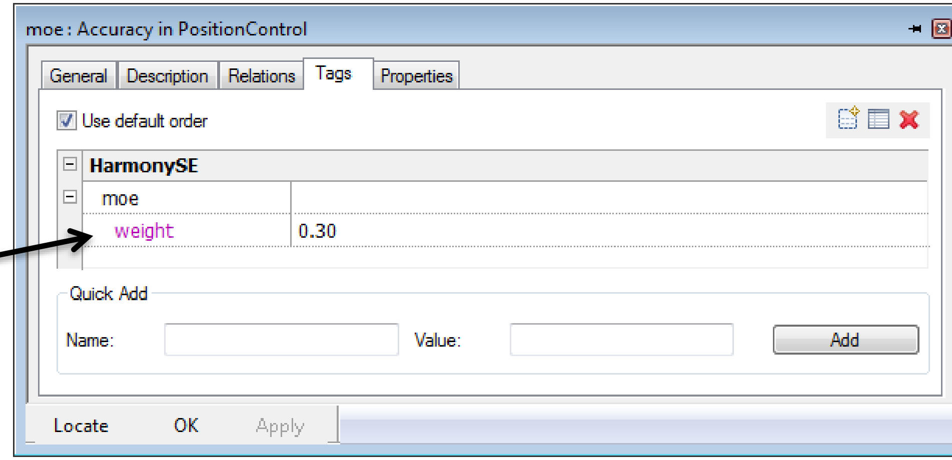
Assign Weights to Assessment Criteria

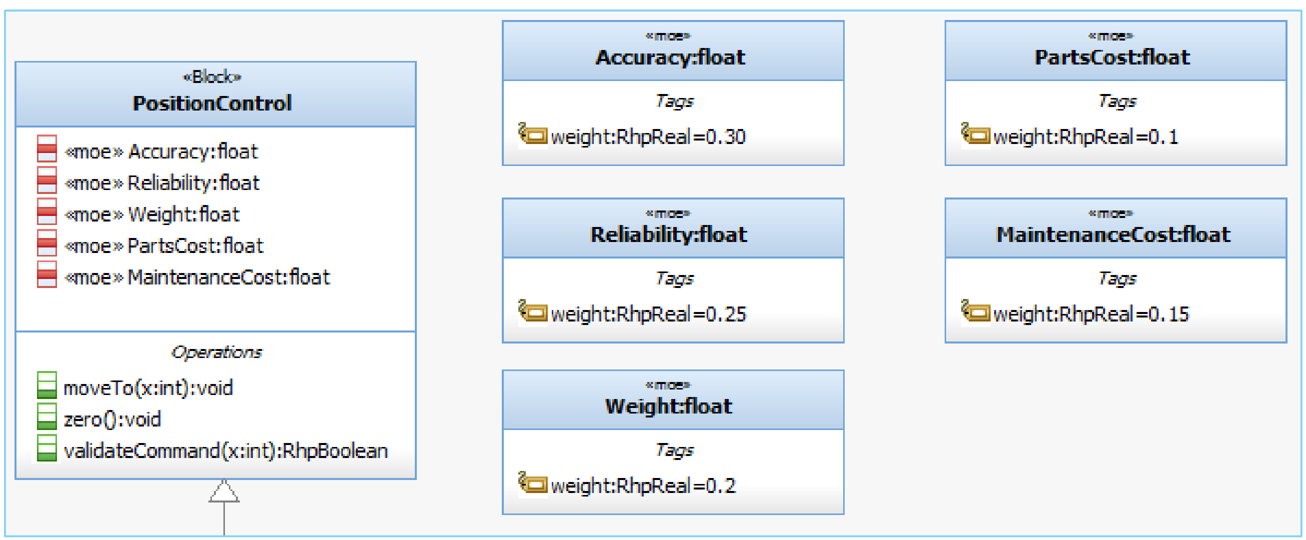
- The weighting value is an assessment of the relative important of that specific criterion to the overall “goodness” of the solution.
  - The higher the weight, the more crucial it is.
  - Normalization (so that the sum of all weights equals 1.00) is a common method use do ensure reasonable relative weighting factors.
- In this case we’ll make the following assignments
  - Accuracy: 0.30
  - Weight: 0.20
  - Reliability: 0.25
  - Parts Cost: 0.10
  - Maintenance Cost: 0.15

Assign Weights to Assessment Criteria

To assign these, double click on each MOE in the browser, go to the *Tags* pane and assign the value:

This is the relative importance of this criteria to the success of the overall solution





Assign Weights to Assessment Criteria

- Tag values are not automatically transferred down the generalization taxonomy
- To copy these down to the children, right click the **PositionControl** block and select *SE-Toolkit > TradeStudies > Copy MOEs to Children*.
  – In this, slightly unusual case, you'll have to repeat the procedure for the **TrimControl** and **ExtensionControl** blocks, as this only works with the immediate children of a block.
  – If you now inspect those subclasses, such as **ElectricTimControl**, you will see that it also has the set of MOEs with the correct values assigned to the weights.

Define Utility Curves

- The utility curve computes a “goodness” score based on a quantitative value associated with the solution.
- The utility curve can be any shape but, by far, those most common is the “linear utility curve.” This curve is a straight line defined by two points.
  - The first point for this MOE is the worst candidate solution being considered has a utility value of 0
  - The best candidate being considered has a value to 10. Given these two points, (worst, 0) and (best, 10), a line can be constructed going through both. This is the linear utility curve.

Define Utility Curves

Since

$$y = \frac{y2 - y1}{(x2 - x1)} x + b$$

We have special conditions, such (worst, 0) and (best, 10) on the line. This simplifies the utility curve to

$$moe = \frac{10}{best - worst} CandidateValue + b$$

And

$$b = -\frac{10}{best - worst} worst$$

Where

- *best* is the value of the criterion for the best candidate solution
- *worst* is the value of the criterion for the worst candidate solution

For example, let's consider a system where our criterion is throughput, measured in messages per second. The worst candidate under consideration has a throughput of 17,000 messages/second and the best candidate has a throughput of 100,000 messages/second. Applying our last two equations provides a solution of

$$moe = \frac{Throughput}{8300} - 170/83$$

A third candidate solution, that has a throughput of 70,000 message per second would then have a computed MOE of 6.3855.

Define Utility Curves

- Get the raw measures of the critera
- Where do these come from?
  - Manufacturer's data
  - Lab measurements
  - Historical data
  - Estimation
  - WAG

Solution/moe	Accuracy (mm)	Weight (kg)	Reliability (mtbf hrs)	Parts cost ($)	Main. Cost ($)
Hydraulic	5	72	4000	800	2000
Electric	1	24	3200	550	2700
Electrohydraulic	2	69	3500	760	2100

Define Utility Curves

- Now develop the utility curves
  - Assume linear unless you have information suggesting non-linear utility curve
  - In this case, linear curves based on (worst, 0) and (best, 10) give the curves:

Solution/moe	Accuracy (mm)	Weight (kg)	Reliability (mtbf hrs)	Parts cost ($)	Main. Cost ($)
Hydraulic	5	72	4000	800	2000
Electric	1	24	3200	550	2700
Electrohydraulic	2	69	3500	760	2100

Using the method outlined above results in the following set of equations:

$$accuracyMOE = -\frac{5}{2}accuracy + \frac{25}{2}$$

$$weightMOE = -\frac{5}{24}weight + 15$$

$$reliabilityMOE = \frac{reliability}{80} - 40$$

$$partCostMOE = -\frac{partsCost}{25} + 32$$

$$maintenanceCostMOE = -\frac{maintenanceCost}{70} + \frac{270}{7}$$

Assign MOEs

- The equations for MOEs can be captured in SysML parametric diagrams.
- In **TrimControlTradeStudy** package and add a new parametric diagram
  - Name this diagram, **Trim Control Trade Study Parametrics**.
- Drag the **PositionControl** block onto the diagram, then drag each of its MOEs to inside the **PositionControl** block on the diagram.
- Add a *ConstraintProperty* from the toolbar onto the diagram. Name this *ConstraintProperty* **TrimControlMOEs**.
- Add ConstraintParameters to the left edge of the constraint property:
  - **accuracy**
  - **weight**
  - **reliability**
  - **partsCost**
  - **maintenanceCost**

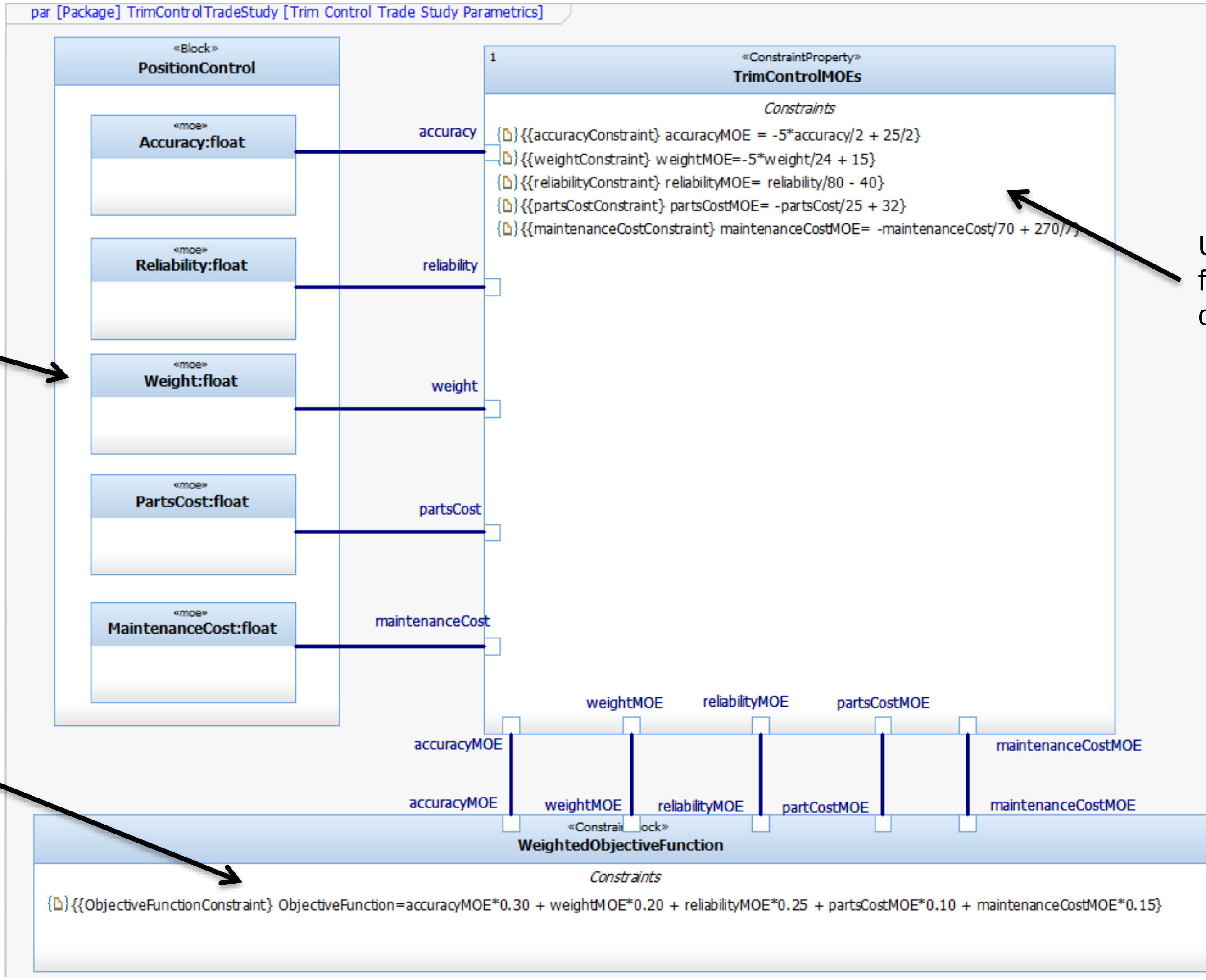
Assign MOEs

- Add *BindingConnectors* between each constraint parameter and the corresponding attribute in the **PositionControl** block.
- Using the technique outlined above, add the equation for each computed MOE, as Constraints in the **TrimControlMOEs** constraint property.
  - **accuracyMOE**
  - **weightMOE**
  - **reliabilityMOE**
  - **partCostMOE**
  - **maintenanceCostMOE**
- Add a new *ConstraintProperty* named **TrimControlObjectiveFunction** and add constraint parameters that match the ones in the previous step
- Add the objective function as a constraint, computing the objective function as the weighted sum of the property times its weighting factor (stored in the **weight** tag)

Assign MOEs

Stand-in for candidate solution

Computation of the overall “”goodness” of the solution



Utility curves for the various design criteria

Assign MOEs: Make a Solution Architecture Diagram

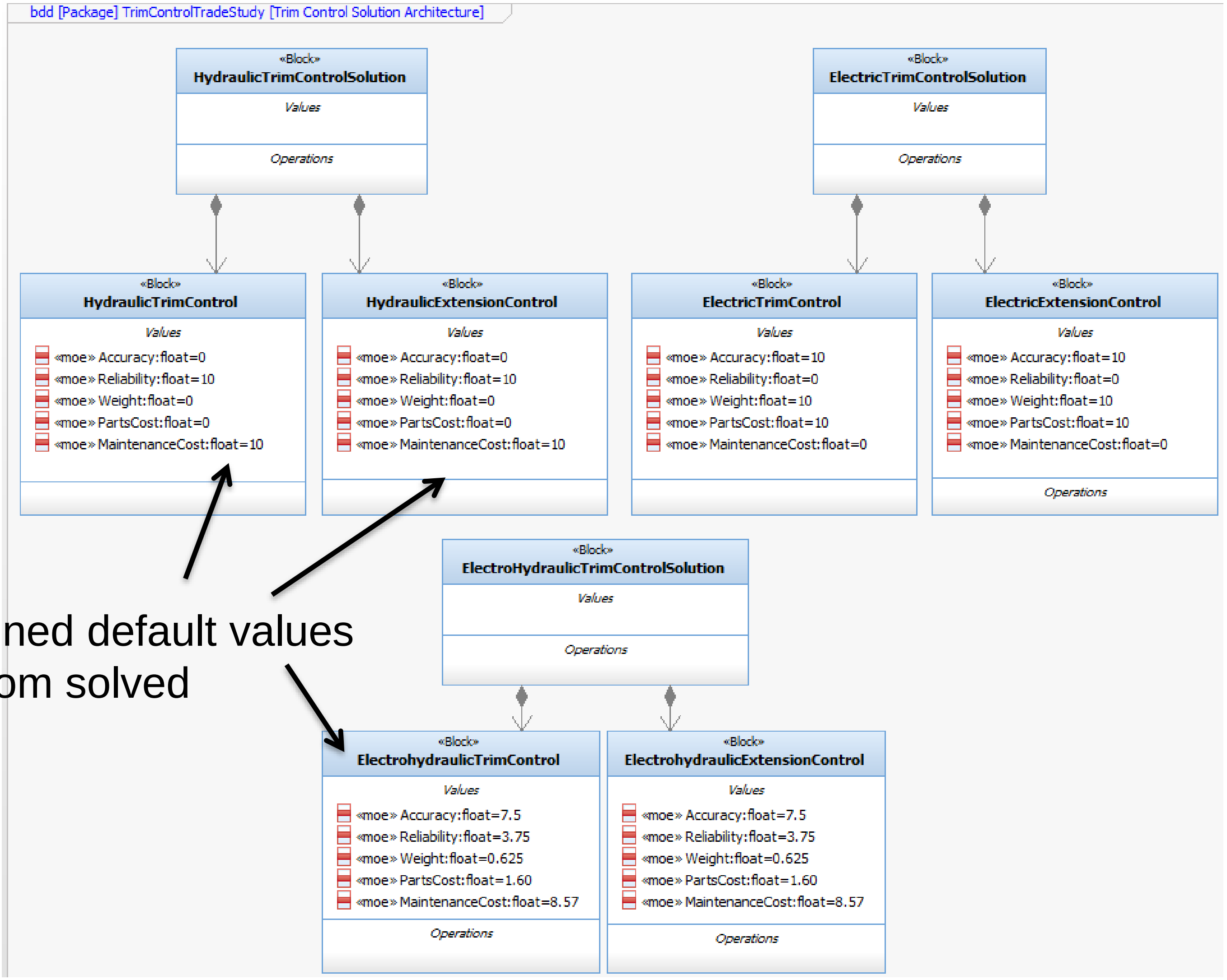
- First, let's build an *Solution Architecture Diagram*. This is a block definition diagram that shows the alternative solutions.
- Add a new Block Definition Diagram named **Trim Control Solution Architecture**.
- Add blocks representing the alternative solution architectures
  - Block **HydraulicTrimControlSolution**
  - Block **ElectricTrimControlSolution**
- Drag the four solution blocks onto the diagram from the browser
  - **HydraulicTrimControl**
  - **HydraulicExtensionControl**
  - **ElectricTrimControl**
  - **ElectricExtensionControl**

Assign MOEs: Make a Solution Architecture Diagram

- Make the appropriate composition relations among the blocks
  - **HydraulicTrimControlSolution** is composed of **HydraulicTrimControl** and **HydraulicExtensionControl**
  - **ElectricTrimControlSolution** is composed of **ElectricTrimControl** and **ElectricExtensionControl**
- Assign the values from solutions values to the *default values* of the appropriate solutions

- Note: Assigning the values for the best and worst scores is easy: it's either 0 or 10, because that's how we defined the linear utility function. To determine the scores are between the best and worst, you'll have to solve the equations above

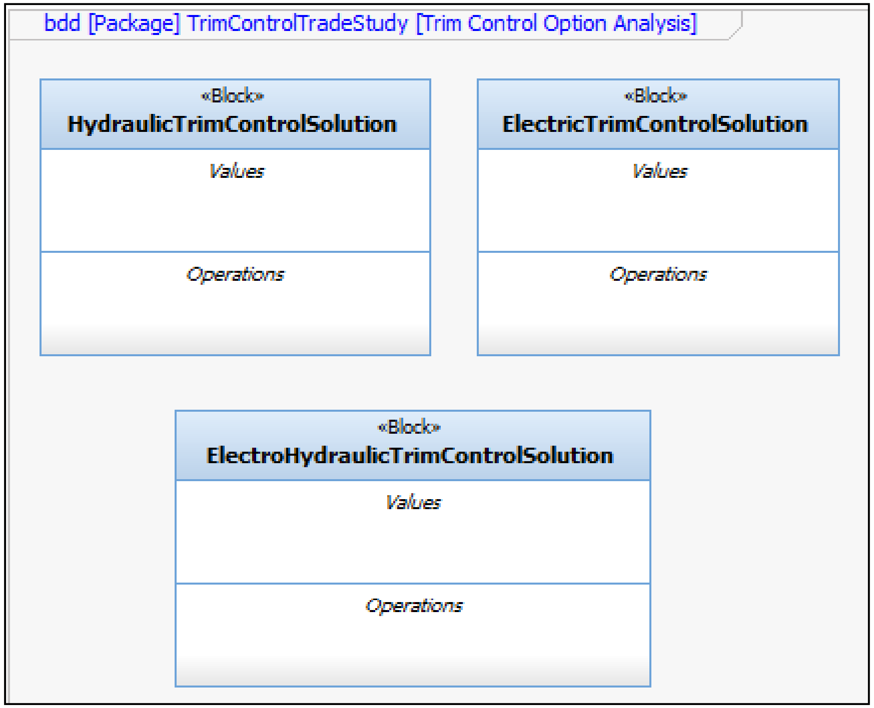
Assign MOEs: Make a Solution Architecture Diagram



Note assigned default values (scores) from solved equations

Determine Best Solution: Make an Option Analysis Diagram

- **Construct an Option Analysis Diagram (BDD)**
- Drag the potential solution architecture blocks on to it: **HydraulicTrimControlSolution, ElectricTrimControlSolution** and **ElectrohydraulicTrimControlSolution.**
- This diagram is very simple and provides a context for the SE-Toolkit to do the analysis:



Determine Best Solution: Make an Option Analysis Diagram

- Right click in this diagram and select *SE-Toolkit > Trade Studies > Perform Trade Analysis*.
  - The toolkit will create a new *Controlled File* named **Trim Control Option Analysis_TradeStudy.xls**.
  - Double-clicking this file will open it in Excel and show you the trade analysis with the computation of the objective function performed by Excel:

		HydraulicTrimControlSolution		ElectricTrimControlSolution		ElectroHydraulicTrimControlSolution	
	weight	value	WV	value	WV	value	WV
PositionControl.Accuracy	0.3	0	0	10	3	7.5	2.25
PositionControl.Reliability	0.25	10	2.5	0	0	3.75	0.9375
PositionControl.Weight	0.2	0	0	10	2	0.625	0.125
PositionControl.PartsCost	0.1	0	0	10	1	1.6	0.16
PositionControl.MaintenanceCost	0.15	10	1.5	0	0	8.57	1.2855
PositionControl.Accuracy	0.3	0	0	10	3	7.5	2.25
PositionControl.Reliability	0.25	10	2.5	0	0	3.75	0.9375
PositionControl.Weight	0.2	0	0	10	2	0.625	0.125
PositionControl.PartsCost	0.1	0	0	10	1	1.6	0.16
PositionControl.MaintenanceCost	0.15	10	1.5	0	0	8.57	1.2855
			8		12		9.516

- By this analysis, the electric motor solution is our best choice, since it has an objective function value of 12, versus 8 for the purely hydraulic solution and 9.516 for the self-contained electrohydraulic units.

Summary: Trade Studies

- The basic workflow is straightforward
  - Identify key system functions that can benefit from optimization
  - Identify candidate solutions
  - Define the assessment criteria
  - Assign weights to criteria (usually normalized)
  - Construct (or estimate) utility curves for each criterion
  - Compute the candidate score for each solution as

$$CandidateScore = \sum C_j W_j$$

  - The solution with the highest score wins
- Block and parametric diagrams are provided in SysML and Rhapsody for representation
- PCE profile allows the evaluation of parametric constraints with third-part math tools using instance specification to provide values for analysis
- Harmony SE-Toolkit provides some automation for evaluation using Excel using default attribute values to provide values for analysis

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

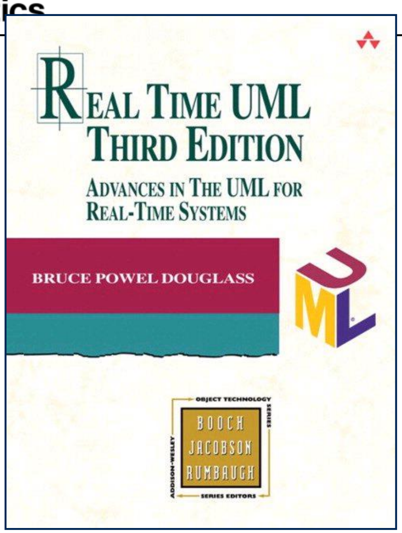
- Developing Embedded Software
- Model-Driven Development for Real-Time Systems
- Model-Based Systems Engineering
- Safety Analysis and Design
- Agile Methods for Embedded Software
- Agile Methods for Systems Engineering
- The Harmony agile Model-Based Systems Engineering process
- The Harmony agile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

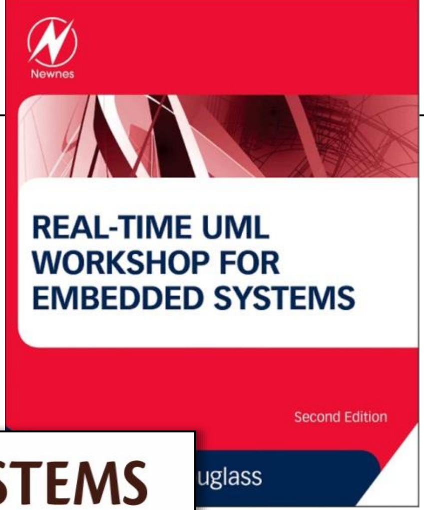
The menu at the top of each page either takes you to the relevant page or to a list of relevant pages.

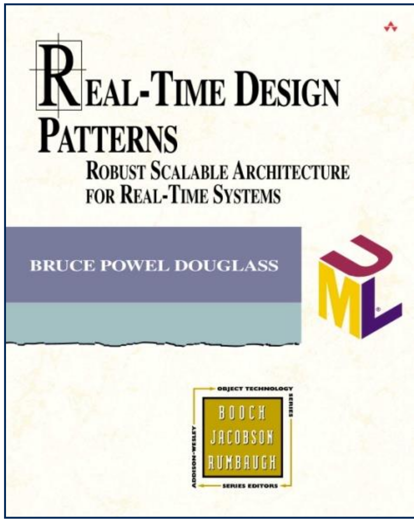
There is even a members only site for those who want to access to even more stuff. I teach and consult on all these topics - see my About page.

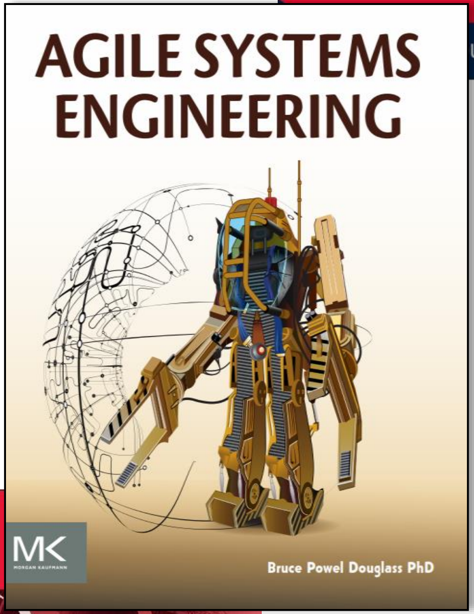
Policy on Using These Materials

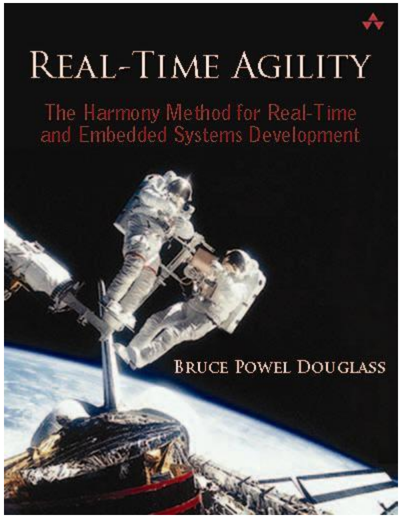
All materials on this web site are free for reuse and distribution, provided their source (me or this web site) is appropriately attributed. I retain sole copyright.

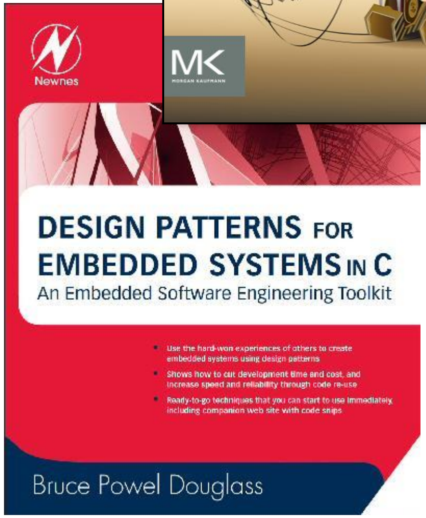

Real Time UML
Third Edition
Advances in The UML for Real-Time Systems
Bruce Powel Douglass
Booch Jacobson Rumbaugh

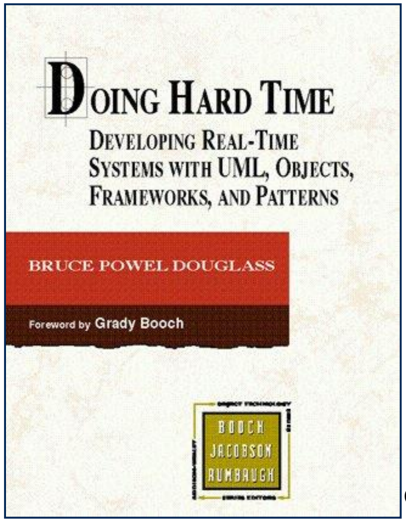

Newnes
REAL-TIME UML WORKSHOP FOR EMBEDDED SYSTEMS
Second Edition
uglass


Real-Time Design Patterns
Robust Scalable Architecture for Real-Time Systems
Bruce Powel Douglass
Booch Jacobson Rumbaugh


AGILE SYSTEMS ENGINEERING
MK
Bruce Powel Douglass PhD


Real-Time Agility
The Harmony Method for Real-Time and Embedded Systems Development
Bruce Powel Douglass


Newnes
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