IBM Research - Haifa

Pluggable Libraries for System Analysis and Optimization

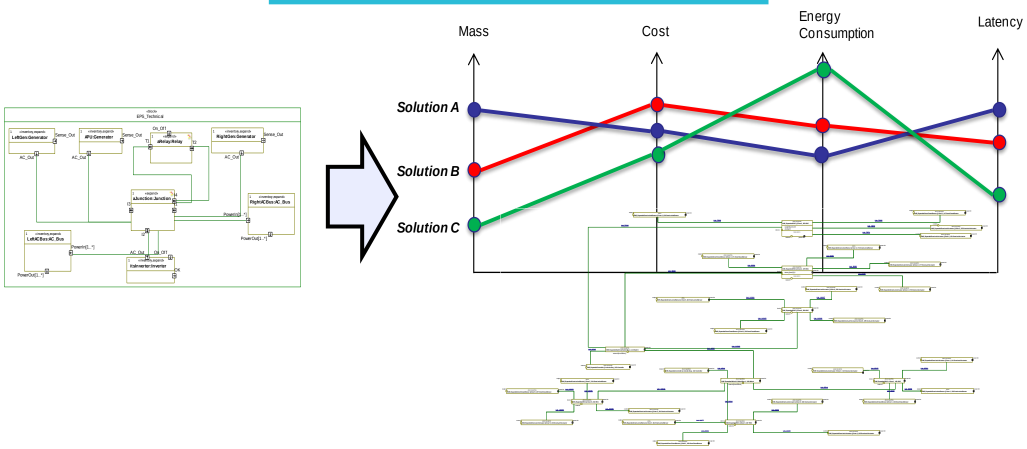
Michael Masin

IBM Research - Haifa

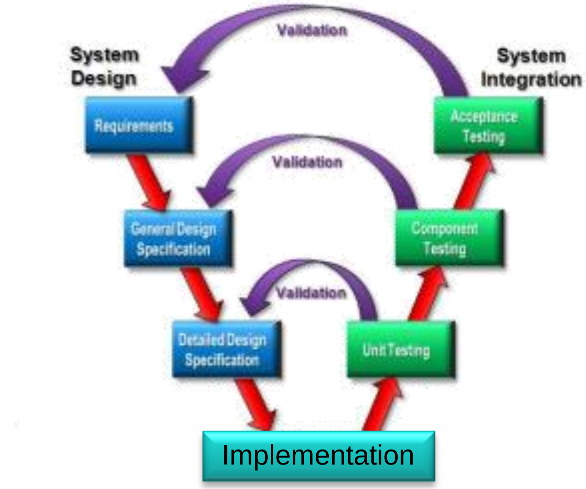
Henry Broodney, Lev Greenberg, Nir Mashkif, Lior Limonad, Aviad Sela, Evgeny Shindin

Systems Engineering research in IBM

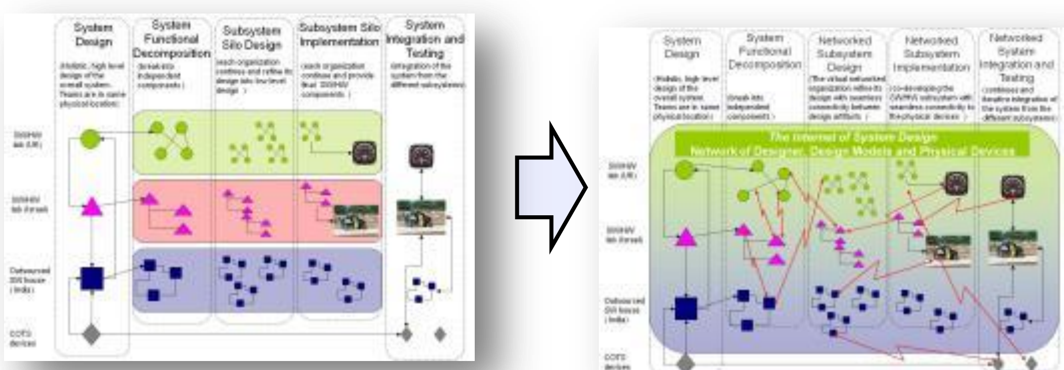
Complex System Optimization



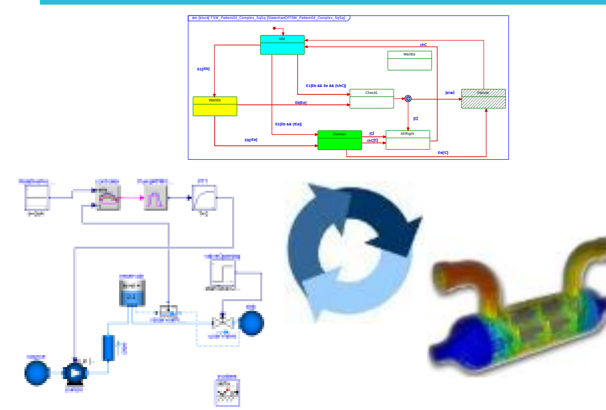
Virtual Verification of Systems



Design Collaboration Frameworks



Complex Hybrid System Simulations



Do not distribute

Agenda

- **Approaches to Architectural Design**
- Background: Architectural Optimization Workbench (AOW)
- Reusable system analysis: objective and use cases
- TotalCost journey
- Paradigm shift: Classification by Property
- Summary

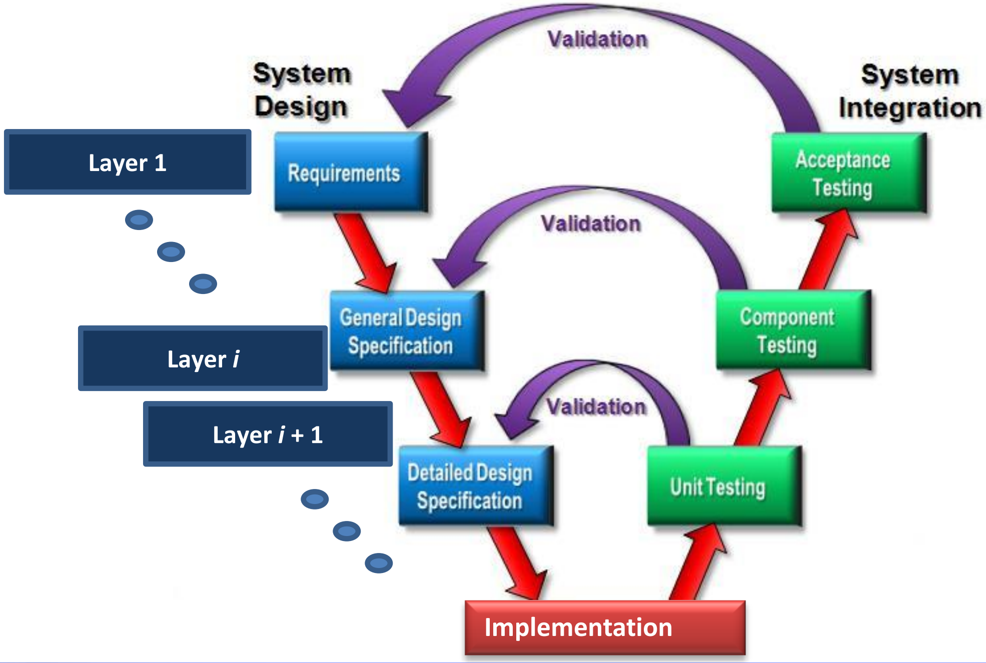
Architectural Design

- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

Architectural Design

- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

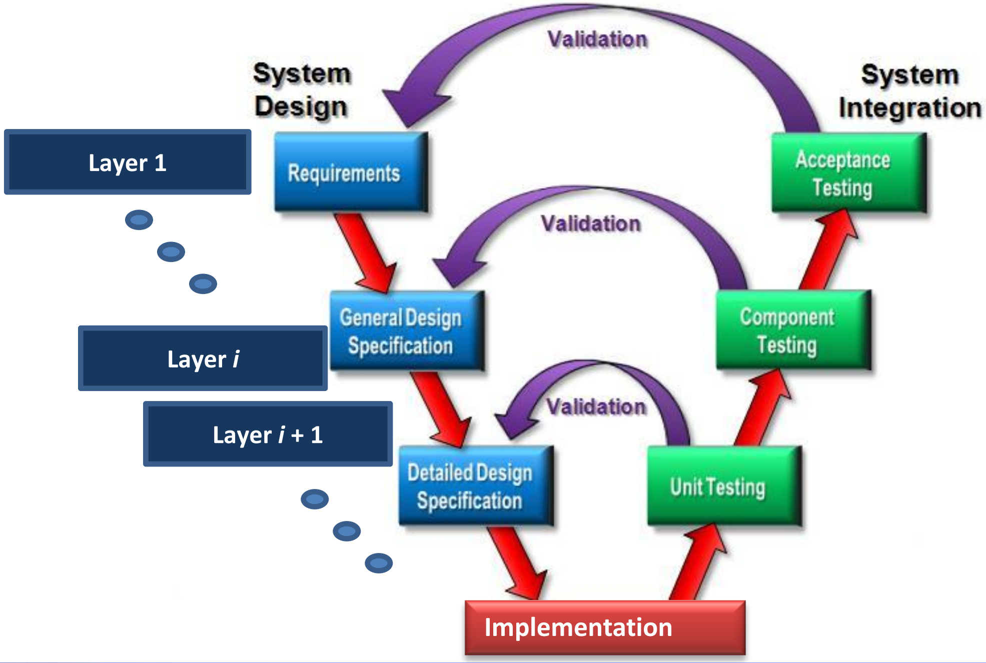
- Solution approaches:
  - Levels of Abstraction



Architectural Design

- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

- Solution approaches:
  - Levels of Abstraction
    - Mapping from Layer *i* (Requirements) to Layer *i*+1 (Architecture)



Architectural Design

- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

- Solution approaches:
  - Levels of Abstraction
  - Separation of concerns

Architectural Design

- Key concern:
  – System Complexity is increasing – manual decisions no longer possible

- Solution approaches:
  – Levels of Abstraction
  – Separation of concerns
    • Multiple viewpoints: functional, technical, geometrical, safety, timing, …
    • Modeling Viewpoints vs. Analysis Viewpoints
    • Independent asynchronous development

Architectural Design

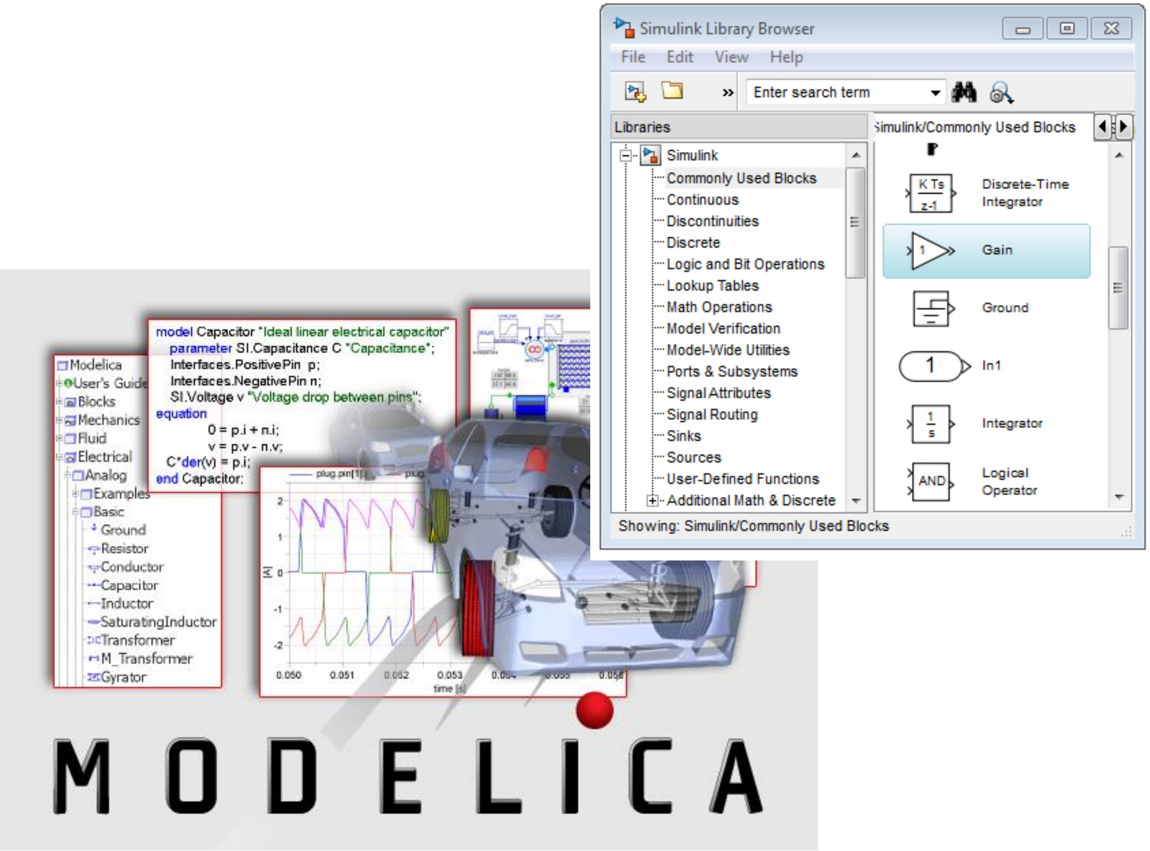
- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

- Solution approaches:
  - Levels of Abstraction
  - Separation of concerns
  - Tools

Architectural Design

- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

- Solution approaches:
  - Levels of Abstraction
  - Separation of concerns
  - Tools
    - Modeling
      - Component Based Design



Architectural Design

- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

- Solution approaches:
  - Levels of Abstraction
  - Separation of concerns
  - Tools
    - Modeling
      - Component Based Design
    - Analysis

Architectural Design

- Key concern:
  - System Complexity is increasing – manual decisions no longer possible

- Solution approaches:
  - Levels of Abstraction
  - Separation of concerns
  - Tools
    - Modeling
      - Component Based Design
    - Analysis
      - Domain specific tools (e.g., SafetyHelper or OpenFTA for safety, SymTA/S for timing)
      - Extension of modeling tools (e.g., PCE in Rhapsody, Simulink Design Optimization toolbox, Optimica extension of Modelica, PaceLab optimization toolkit)
      - Black box integration (e.g., ModelCenter, modeFrontier)
      - Custom optimization modeling (in AMPL, OPL, GAMS, AIMMS)

Objective

1. Talking Systems Engineers language, applying the best optimization tools

Agenda

- Approaches to Architectural Design
- **Background: Architectural Optimization Workbench (AOW)**
- Reusable system analysis: objective and use cases
- TotalCost journey
- Paradigm shift: Classification by Property
- Summary

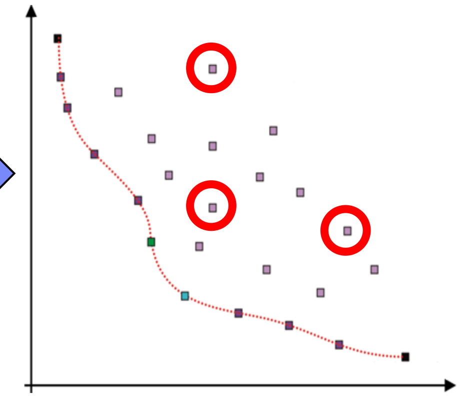
AOW Vision: Accelerate Smarter Architectural Decisions

NOW

Definition of design alternatives

Analysis

Review



- Manual process
  - Mostly based on individual undocumented experience
  - Lots of reuse of previous designs
  - No assurance for solution optimality
  - Each additional design – linear effort
- No formal capture of designs
  - Modeling for specific analysis types

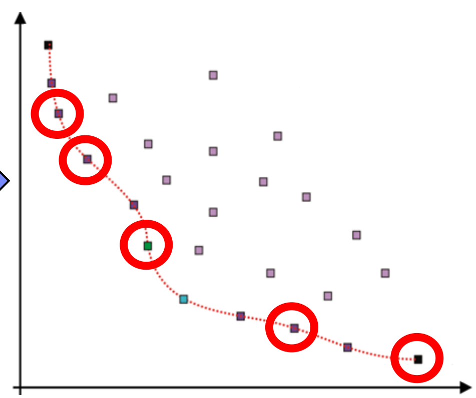
time

With AOW

Architectural template definition

Review

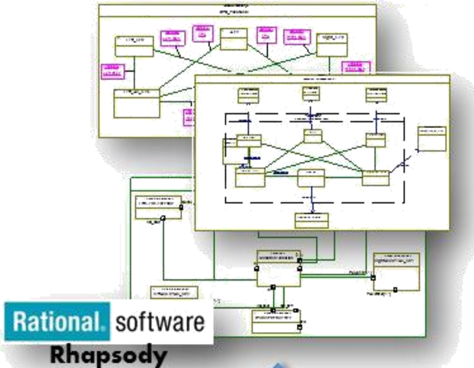
Automated Design Space Exploration



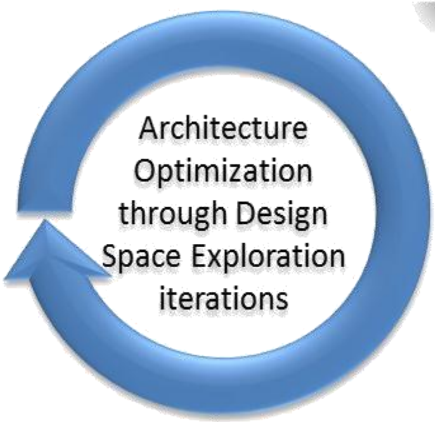
- Formal composition rules and constraints
- Automatic generation of optimization code
- Transparency, traceability, visualization of results
- Knowledge capture and reuse
- Optimization Capabilities in the hands of Architects
- Management of tools orchestration and collaboration

IBM's Architecture Optimization Workbench Concept

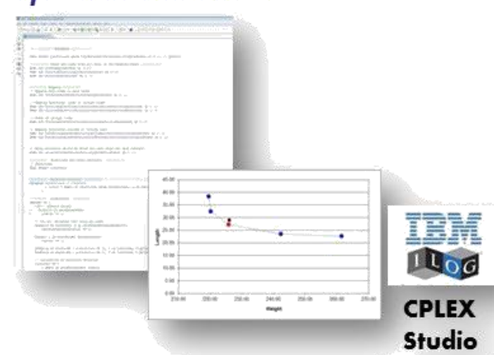
1. Describe system through different SysML views, including design alternatives, constraints and goals



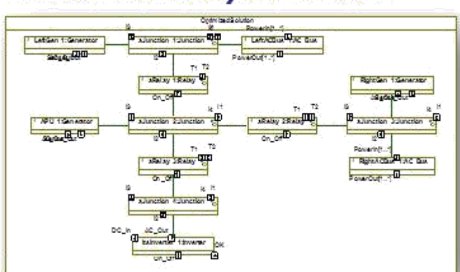
2. Derived Data Schema for Input and output structures



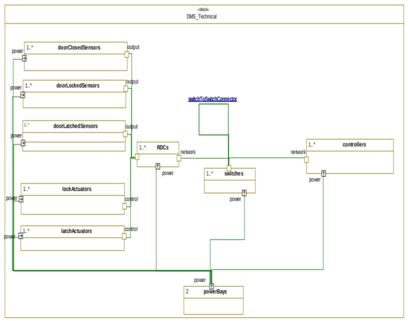
3. Automatic translation(via an interchange format) into Optimization solver

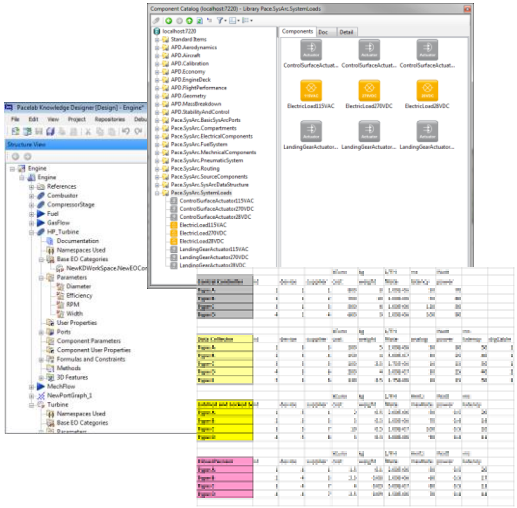


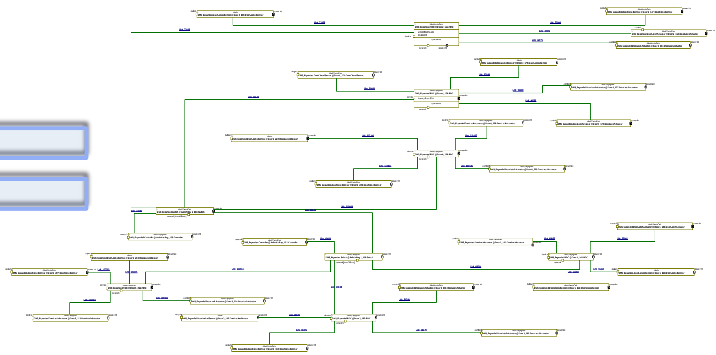
4. Optimized architecture back annotated to SysML model



AOW uses Concise Modeling







- Large systems architectures are difficult to model
  - Lots of elements and details
  - Time consuming
  - Error prone
- Concise Modeling – SysML models combined with tabular data
  - SysML depicts the system composition rules
  - Tables contain instantiations
- For an existing model the tabular data is taken from a component library
- In AOW some tables are filled by the optimization engine

Optimization Auto-Generation

▪*Requirements & Constraints*

P1: whenever [E] occurs [SC] holds during following [I]

P2: whenever [E1] occurs [E2] implies [E3] during following [I]

P3: whenever [E1] occurs [E2] does not occur during following [I]

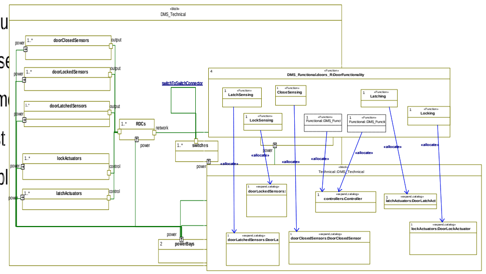
P4: whenever [E] occu

P5: [SC] during [I] raise

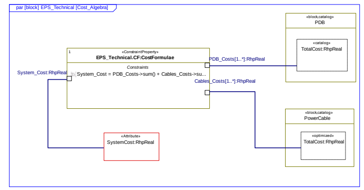
P6: [E1] occurs [N] tim

P7: [E] occurs at most

P8: [SC] during [I] impl



▪*Objectives*



▪*Component library*

▪Automatic Generation

▪*Optimization Problem Formulation (CPLEX)*

```
minimize weightPriority * totalWeight + lengthPriority * totalLength;


////////////////////  Constraints //////////////////////////////////////

subject to {

//
//// Constraint to use Place FA, e.g., according the System Eng suggestion for a good weight solution
//
//  UseFAConst:
//      forall(j in places : j.id == 1) usePlace[j] == 1;

//========   Objective metrics   ========

  TotalWeightConst:
    totalWeight == sum(j in places, p in pdbs) pdb4place[p][j] * p.weight
        + sum(j in places, w in wires) wire2place[w][j] * w.weightPerMeter * j.fromBus
        + sum(i in loads, j in places, l2p in loads2places : l2p.load == i.id && l2p.place == j.id, w in wires)
            wire2load[w][j][i] * w.weightPerMeter * l2p.dist
        ;

  TotalLengthConst:
    totalLength == sum(j in places) usePlace[j] * j.fromBus
        + sum(i in loads, j in places, l2p in loads2places : l2p.load == i.id && l2p.place == j.id)
            place4load[j][i] * l2p.dist
        ;

//========   Mapping contracts   ========

// Geometrical to functional (pdb placeholder per functional link)
  OnePlace4LoadConst:
    forall(i in loads)  sum(j in places) place4load[j][i] == 1;

// Technical to geometrical (pdb to pdb placeholder)
  UptoOnePDB4placeConst:
    forall(j in places)  sum(p in pdbs) pdb4place[p][j] == usePlace[j];

// Technical to geometrical (wire to pdb-bus link)
  UptoOneWire2placeConst:
    forall(j in places) sum(w in wires) wire2place[w][j] == usePlace[j];

// Technical to geometrical (wire to pdb-load link)
  UptoOneWire2LoadConst:
    forall(i in loads, j in places) sum(w in wires) wire2load[w][j][i] == place4load[j][i];


//========   Network contracts   ========

// If pdb placeholder is used for any load, it should be used
  UsedPlaceConst:
    forall(i in loads, j in places)  usePlace[j] >= place4load[j][i];

// If pdb placeholder is NOT used for any load, it should NOT be used (could be removed because of Objective function)
  NotUsedPlaceConst:
    forall(j in places)  usePlace[j] <= sum(i in loads) place4load[j][i];

```

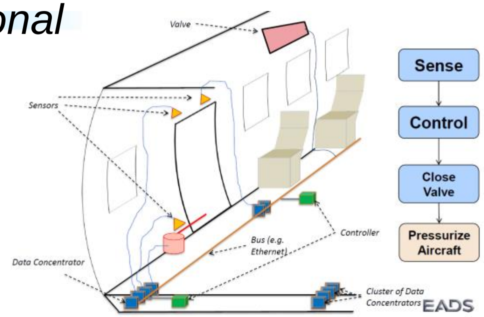
EADS, Doors Management System

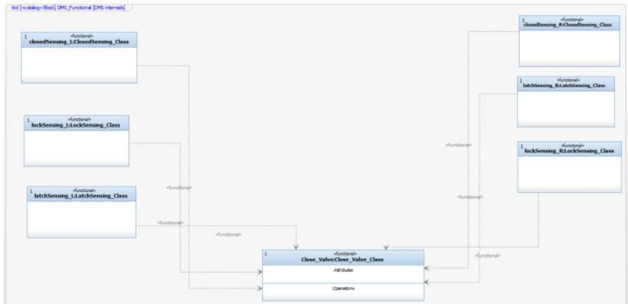
Development and Analysis of a Simplified Doors Control System

- **Monitor and Control Passenger Doors, Emergency Exits, and Cargo Doors**
- **Design a system out of existing components for best weight, cost, power etc.**

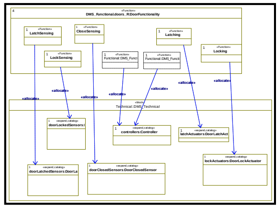
DMS: From user story to model

Functional

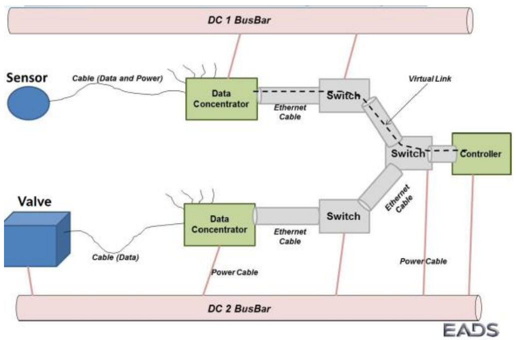


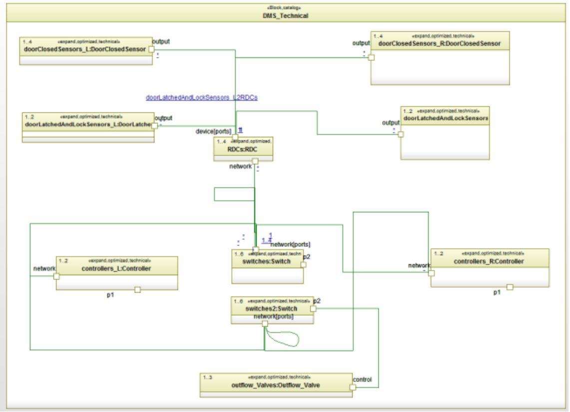


Mapping

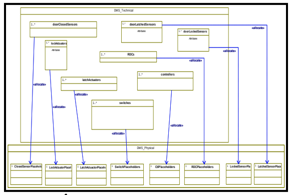


Technical

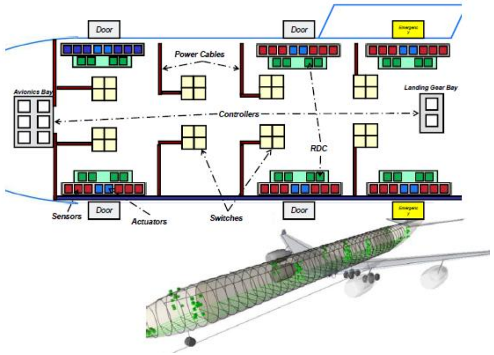


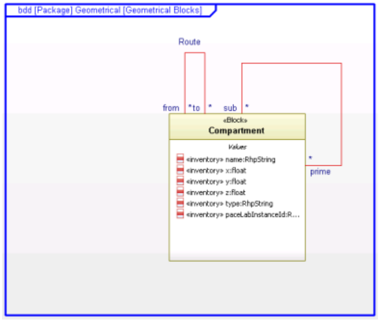


Allocation



Geometrical

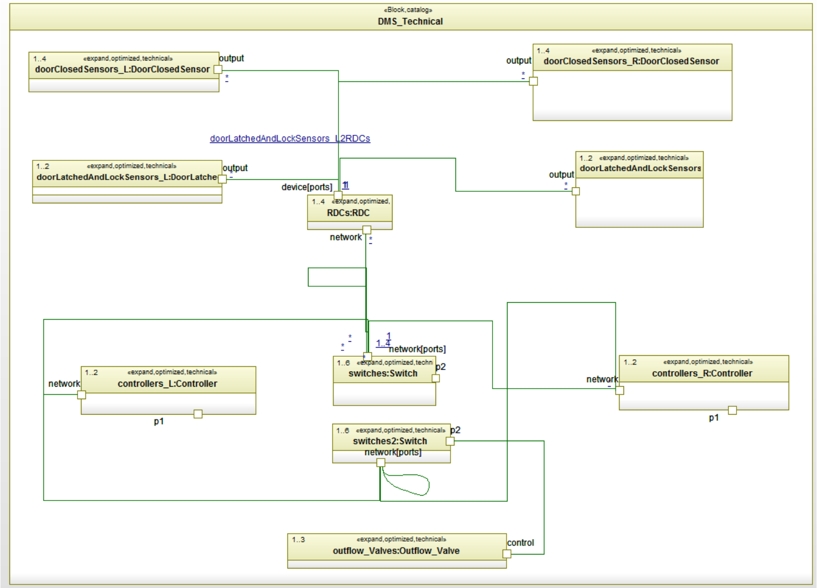




Do not distribute

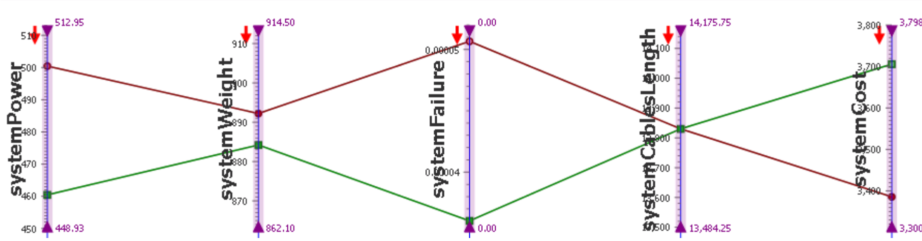
DMS: Results [1/2]

1. SysML modeling

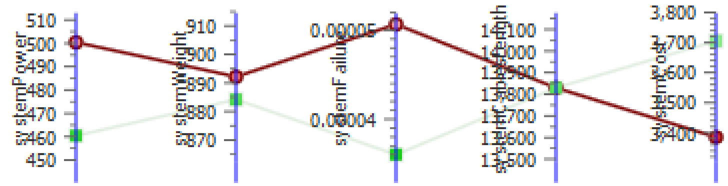


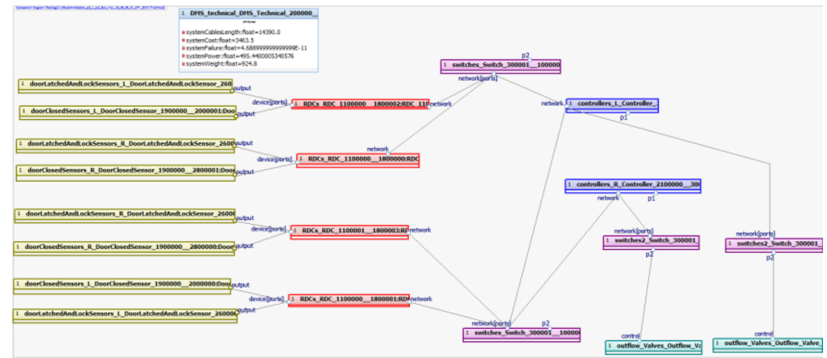
2. Run Optimization and show alternatives

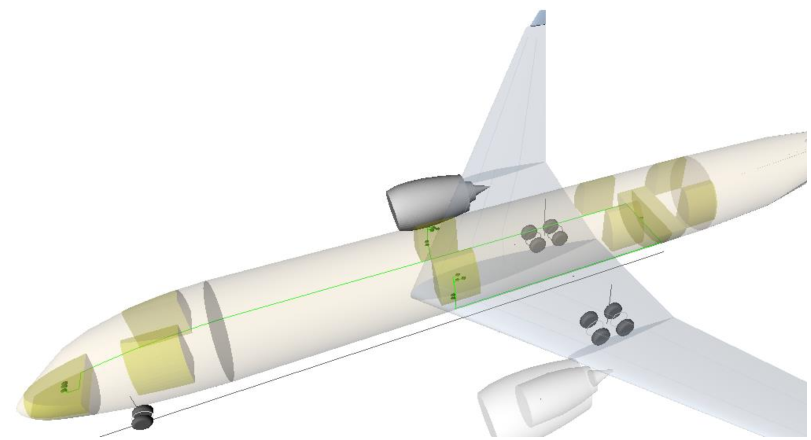
ID	systemPower	systemWeight	systemFailure	systemCablesLength	systemCost	Finished at	Duration
2ece937c-a4b9-4ac8-b0ab-702de3a1c94a	500.44	892.20	5.066E-5	13830.00	3385.50	May 23, 2013 11:14:54 AM	942 sec
3232cbf2-0383-4500-811a-52d643079cd3	460.44	884.20	3.602E-5	13830.00	3705.50	May 23, 2013 11:10:54 AM	938 sec



3. Visualize the alternatives

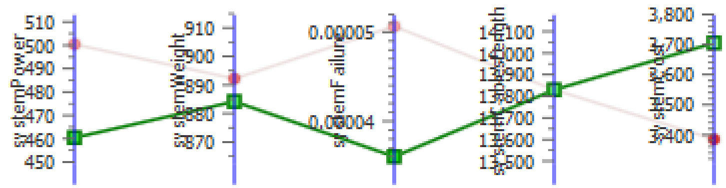


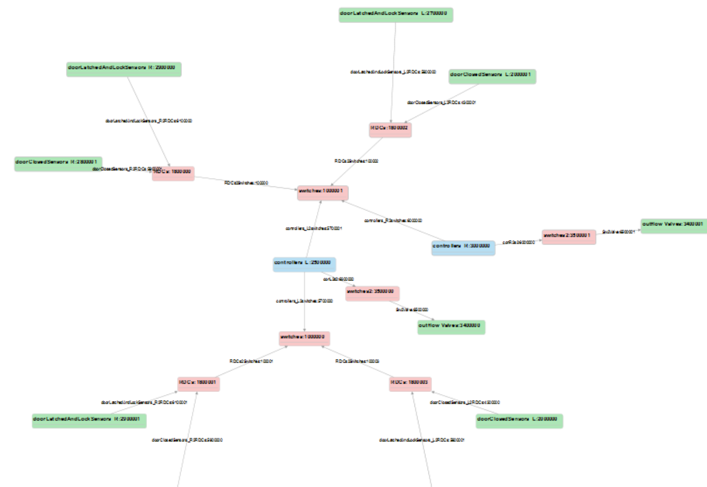




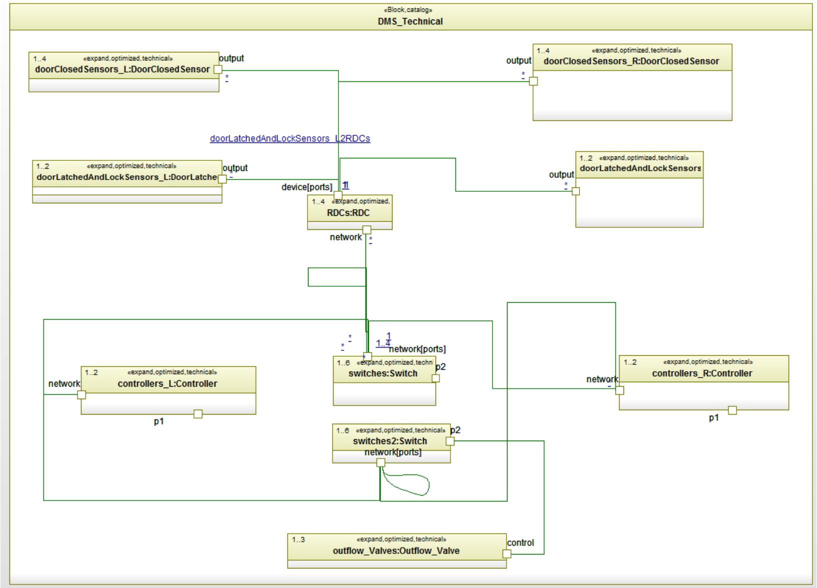
DMS: Results [2/2]

3. Visualize the alternatives



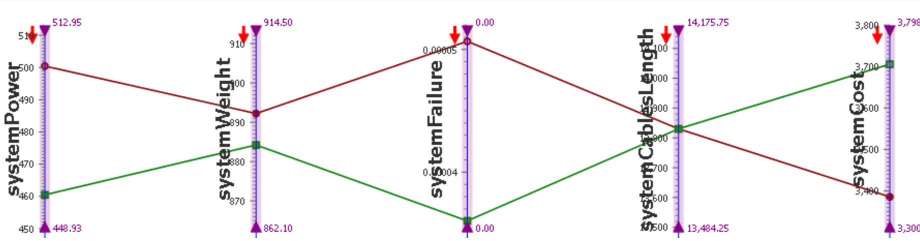


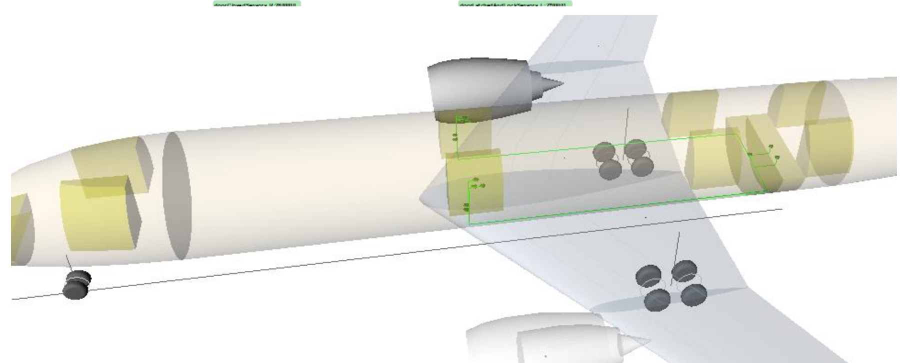
1. SysML modeling



2. Run Optimization and show alternatives

ID	systemPower	systemWeight	systemFailure	systemCablesLength	systemCost	Finished at	Duration
2ece937c-a4b9-4ac8-b0ab-702de3a1c94a	500.44	892.20	5.066E-5	13830.00	3385.50	May 23, 2013 11:14:54 AM	942 sec
3232cbf2-0383-4500-811a-52d643079cd3	460.44	884.20	3.602E-5	13830.00	3705.50	May 23, 2013 11:10:54 AM	938 sec





Agenda

- Approaches to Architectural Design
- Background: Architectural Optimization Workbench (AOW)
- **Reusable system analysis: objective and use cases**
- TotalCost journey
- Paradigm shift: Classification by Property
- Summary

Objective

1. Talking Systems Engineers language, applying the best optimization tools
2. **Library of reusable pluggable Analysis Viewpoints**

Objective

1. Talking Systems Engineers language, applying the best optimization tools
2. **Library of reusable pluggable Analysis Viewpoints**

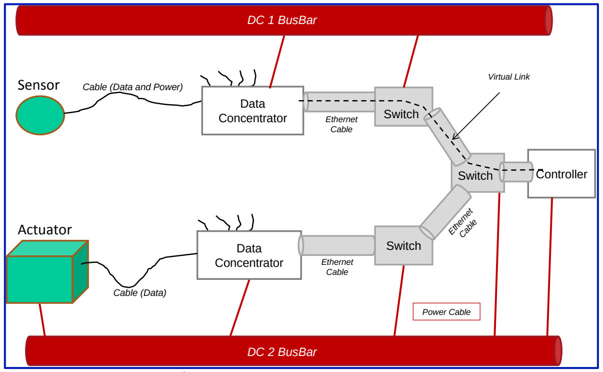
Questions

- Do we really have repeating types of analysis?
- Does current optimization modeling support building the libraries?
- What should be the methodology and supportive framework?

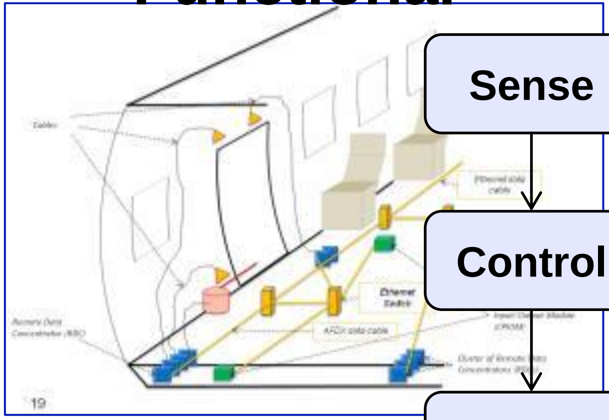
Use Case 1 – EADS
Development and Analysis of a Doors and Slides Control System

- **Passenger Doors, Emergency Exits, and Cargo Doors**

Structure



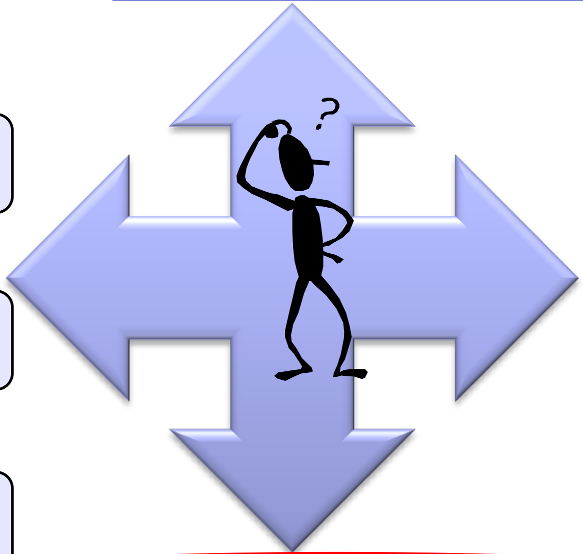
Functional



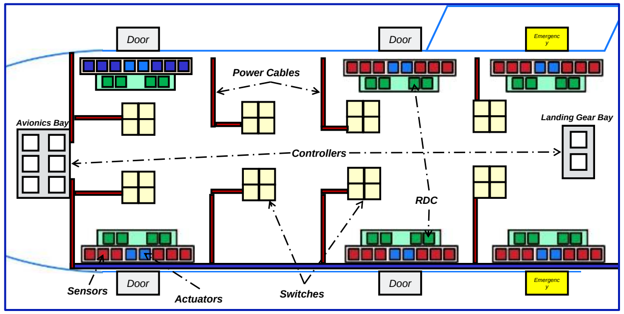
Sense

Control

Actuate



Geometrical

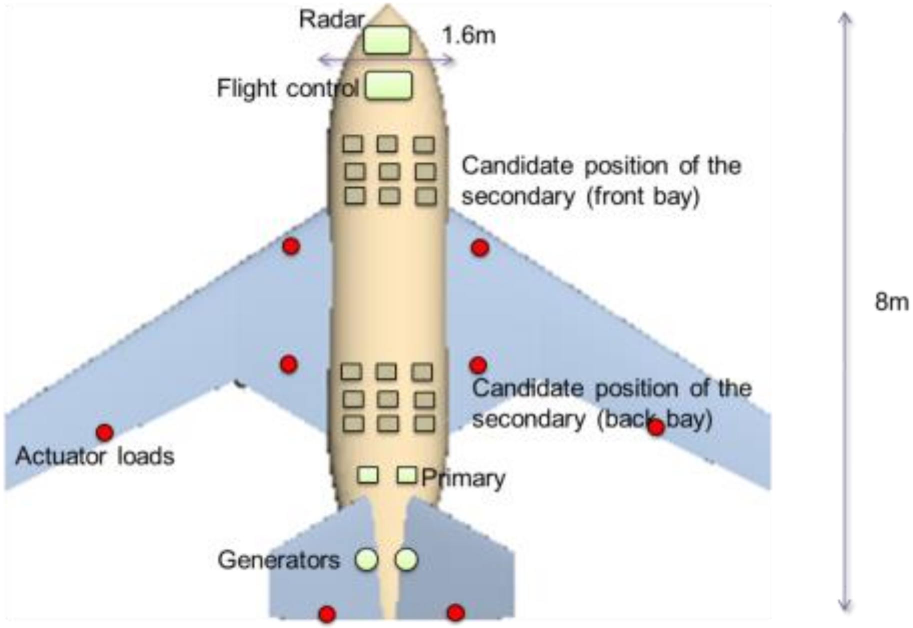


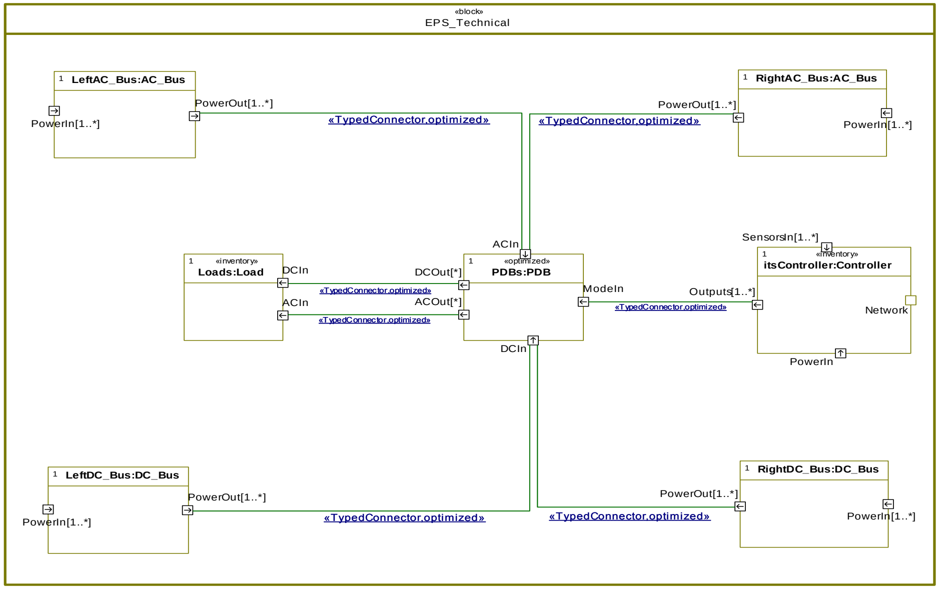
Design metrics

Weight, Cable length, Power distribution, Mapping, Allocation, Reliability …

Use Case 2 – UTC
Power Distribution System

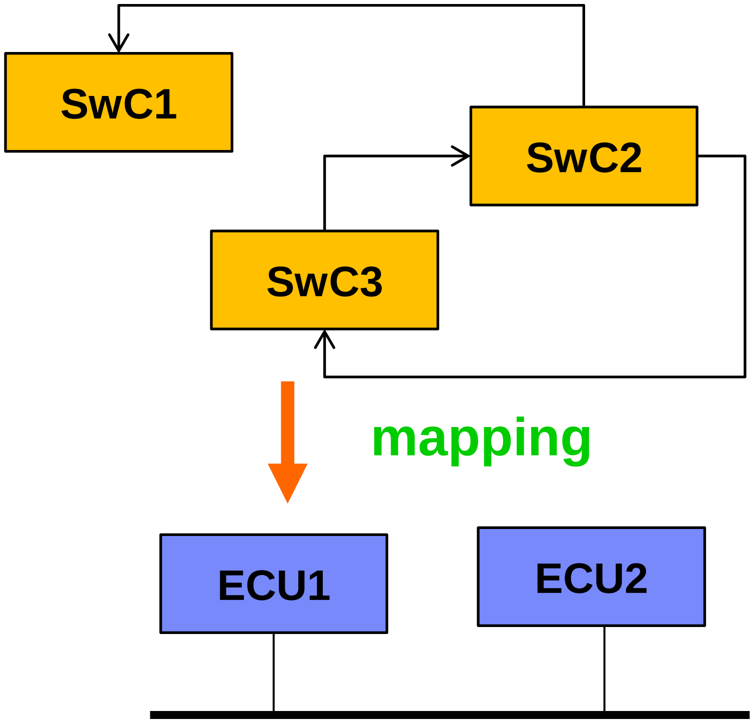
- Typical Power Distribution System
  - Draw power from sources and distribute to the loads
  - The core is a collection of Power Distribution Boxes (PDB) housing electric protection and power management circuitry
  - Several loads per PDB
- Task - optimal selection, allocation, connections, and placement of PDBs
- Inputs:
  - Loads' power requirements
  - Available components properties
  - Geometry of the platform
- Design metrics:
  - Mapping
  - Weight
  - Cost
  - Cable length
  - Reliability requirements
  - Power efficiency





Use Case 3 – Automotive

- Complex system topology
- Optimal mapping of SW components to ECUs



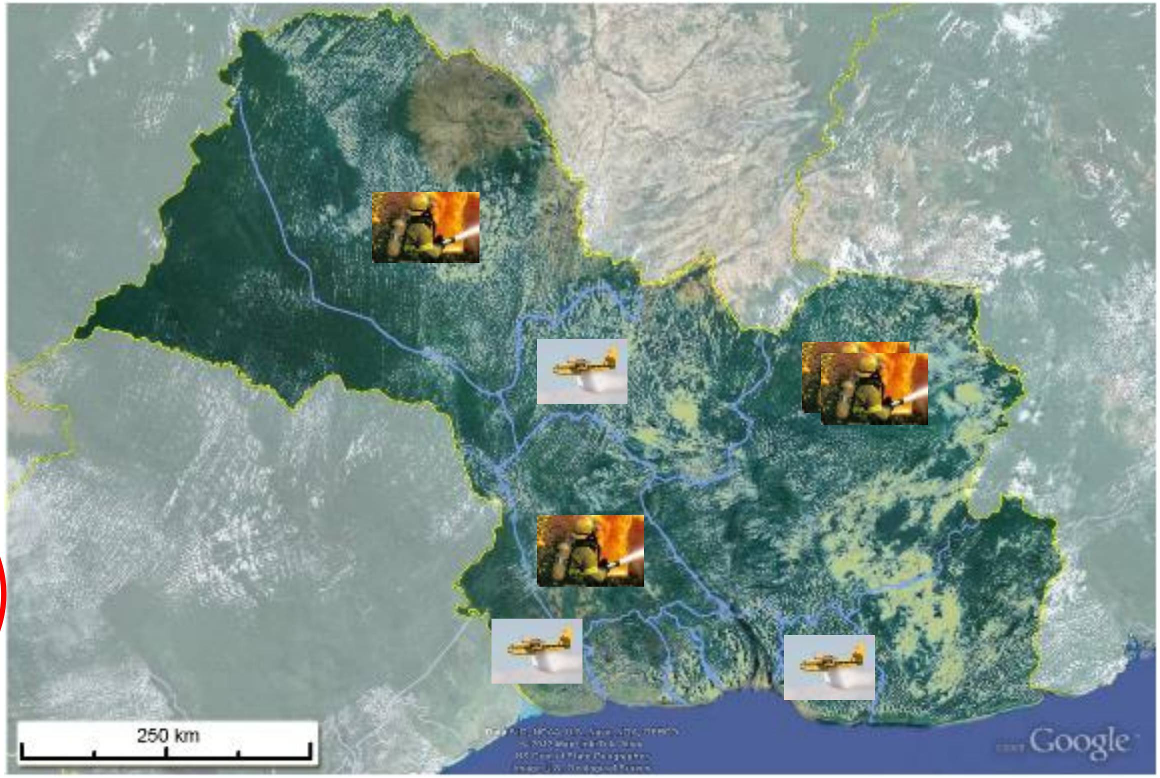
Design metrics:

- Mapping
- Reliability
- Cost
- Power
- Timing/schedulability
- Data communication
- Operational scenarios

Use Case 4 - INCOSE 2012 Tool Vendor Challenge Forest Fire Fighting

- A country with several districts has a history of forest fires. Each district has a fire detection and fire fighting forces and there is no cooperation between them. The government wants to build a Command and Control center that will coordinate between the forces. The government wants to know how much savings this can bring, to justify the cost of the C&C.

- Task - optimal placement of detection and firefighting assets in the districts so that all the required scenarios are successfully dealt with.

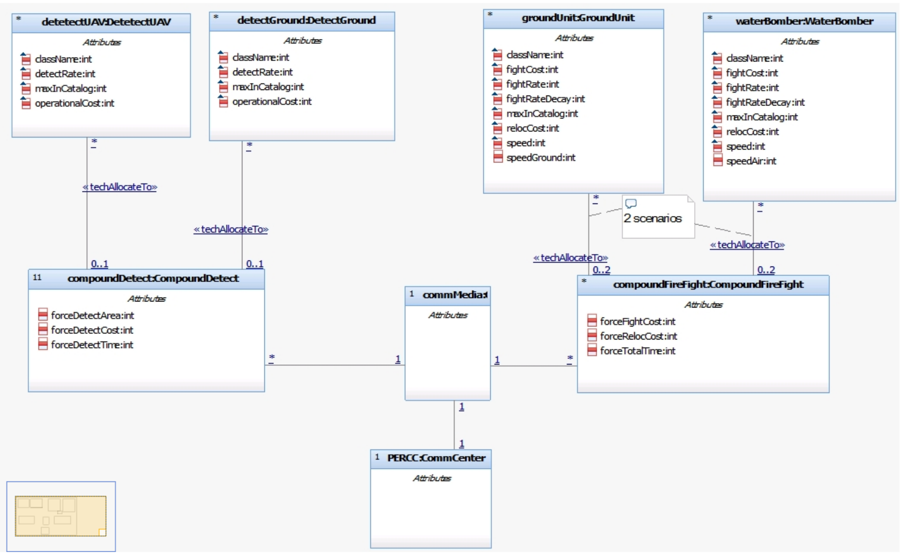


Design metrics:

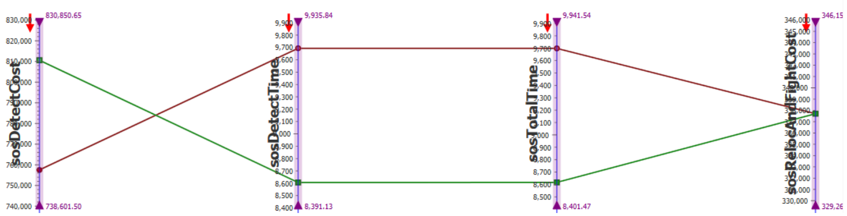
- Resource allocation
- Cost
- Timing
- Data communication
- Scenarios

INCOSE 2012 Tool Vendor Challenge - Forest Fire Fighting, Results

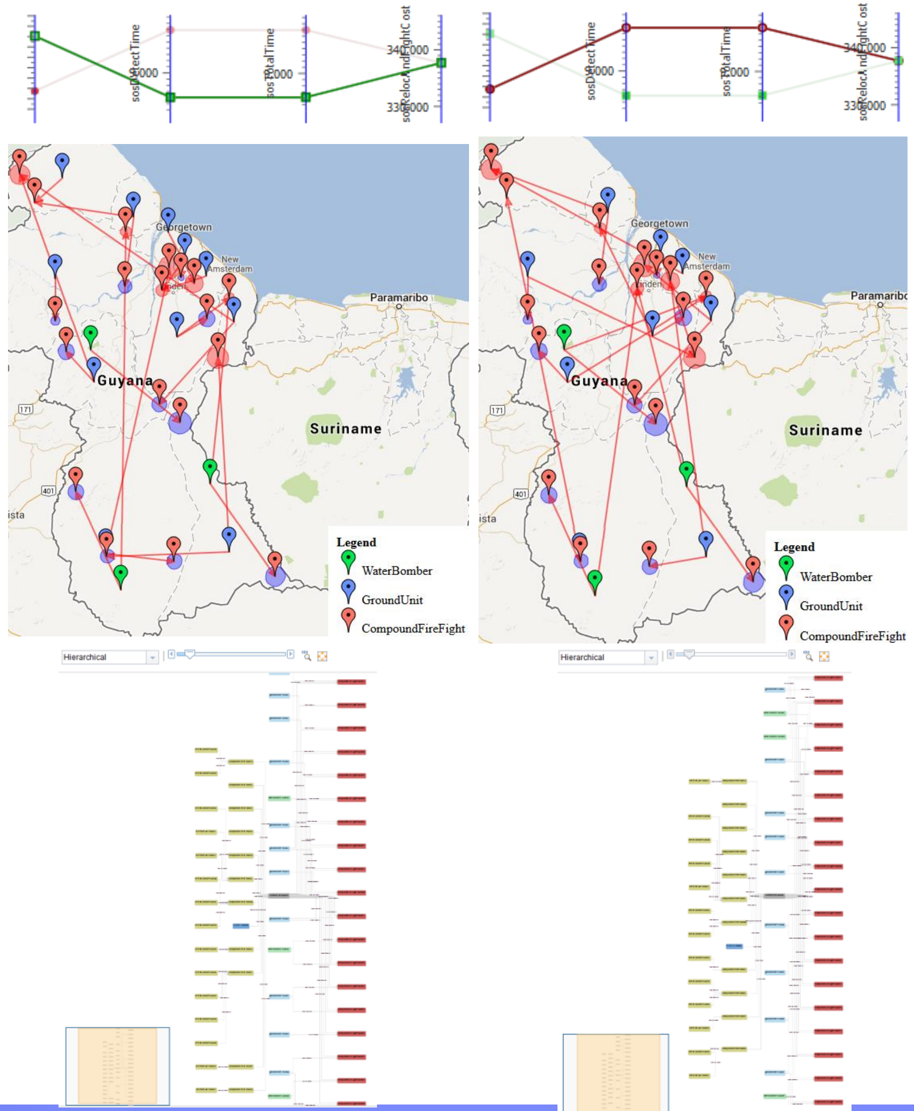
1. SysML modeling



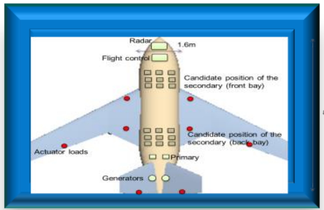
2. Run Optimization and show alternatives



3.Visualize the alternatives

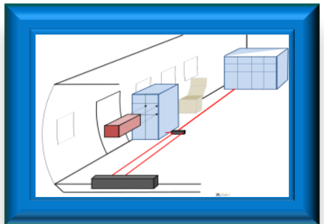


Use cases
we have explored so far



UTC, Power Distribution System

Draw power from sources and distribute to the loads and PBC, optimal selection, allocation, connections, and placement

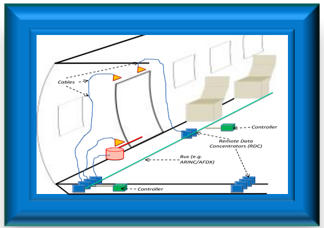


Airbus, Cabin Configuration

Find Optimal Cabin Architecture fulfilling different airlines requirements by Minimize cost, weight, power and voltage drop, and redundancy

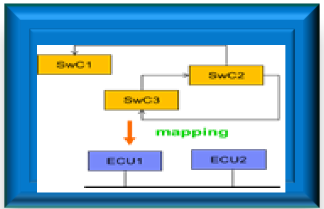
INCOSE 2012, Forest Fire Fighting

Optimal placement of detection and firefighting assets in the districts so that all the required scenarios are successfully dealt with



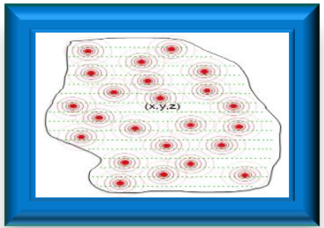
EADS, Doors Management System

Find Optimal architecture, and geo allocation considering cost, cable length, weight and reliability aspects.



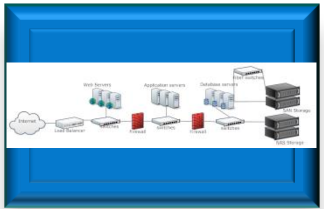
Automotive, E/E usecase

T1/T2 suppliers provide software modules with their hardware, OEMs are required to allocate each module to an ECU



DANSE, Sensor positioning in System of Systems context

Assign the possible antenna bridge positions , Estimate the coverage area for each antenna



IT Architecture design

Build IT architecture based on applications requirements, interconnection, application, software and hardware catalogues

Objective

1. Talking Systems Engineers language, applying the best optimization tools
2. Library of reusable pluggable Analysis Viewpoints

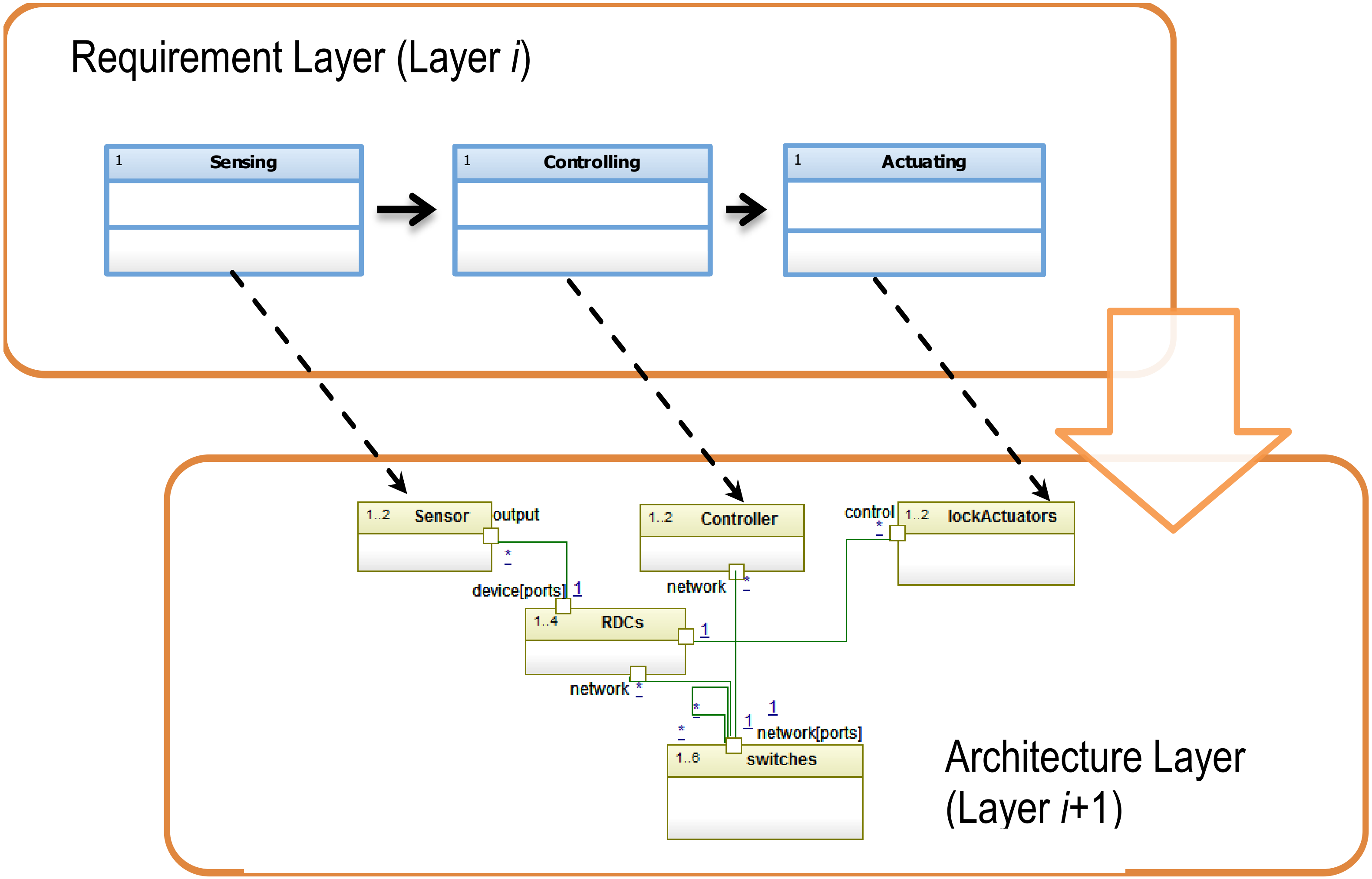
Questions

- √ Do we really have repeating types of analysis?
- Does current optimization modeling support building the libraries?
- What should be the methodology and supportive framework?

Agenda

- Approaches to Architectural Design
- Background: Architectural Optimization Workbench (AOW)
- Reusable system analysis: objective and use cases
- ***TotalCost* journey**
- Paradigm shift: Classification by Property
- Summary

Example



Requirement Layer (Layer i)
1
Sensing
1
Controlling
1
Actuating
1..2
Sensor
output
*
device[ports]
1
1..2
Controller
network
*
control
*
1..2
lockActuators
1..4
RDCs
1
network
*
*
*
1
1
network[ports]
1..6
switches
Architecture Layer
(Layer i+1)

totalCost Journey

Minimize totalCost

Subject to

$$totalCost = \sum_{j \in SensorTypes} \sum_{i \in Sensors} SensorType[j].Cost \cdot sensor[i][j] + \cdots$$

$$+ \sum_{j \in SwitchTypes} \sum_{i \in Switches} SwitchType[j].Cost \cdot switch[i][j]$$

…

// Mapping

$$\sum_{j \in SensorTypes} \sum_{i \in Sensors} sensing2sensor[l][i][j] = 1 \quad \forall\, l \in SensingFunctions$$

totalCost Journey

- Requirement for using several types of sensors
  - thermal and volume sensors
  - each having its own attributes and a catalog of available types

$$totalCost =$$

$$\sum_{j \in TermalSensorTypes} \sum_{i \in TermalSensors} ThermalSensorType[j].Cost \cdot thermalSensor[i][j]$$

$$+ \sum_{j \in VolumeSensorTypes} \sum_{i \in TermalSensors} VolumeSensorType[j].Cost \cdot volumeSensor[i][j] + \cdots$$

$$+ \sum_{j \in SwitchTypes} \sum_{i \in Switches} SwitchType[j].Cost \cdot switch[i][j].$$

totalCost Journey

- Cable component connects other architectural components
  - new type – Cable
  - cables' costs are also dependent on the geometrical layout

$$totalCost =$$

$$\sum_{j \in SensorTypes} \sum_{i \in Sensors} \sum_{\boldsymbol{k} \in \boldsymbol{Compartments}} SensorType[j].Cost \cdot sensor[i][j][\boldsymbol{k}] + \cdots$$

$$+ \sum_{j \in CableTypes} \sum_{i \in Cables} \sum_{\boldsymbol{k} \in \boldsymbol{SPaths}} CableType[j].Cost \cdot cable[i][j][\boldsymbol{k}] .$$

totalCost Journey

- Safety requirements
  - functions and functional links could be implemented by several redundant channels

$$\sum_{j \in SensorTypes} \sum_{i \in Sensors} sensing2sensor[l][\boldsymbol{r}][i][j] = redundancyChannel[l][r]$$

$$\forall\ l \in SensingFunctions, \boldsymbol{r} \in \boldsymbol{RedundancyChannels}$$

Objective

1. Talking Systems Engineers language, applying the best optimization tools
2. Library of reusable pluggable Analysis Viewpoints

Questions

√ ▪ Do we really have repeating types of analysis?

✗ ▪ Does current optimization modeling support building the libraries?

▪ What should be the methodology and supportive framework?

Agenda

- Approaches to Architectural Design
- Background: Architectural Optimization Workbench (AOW)
- Reusable system analysis: objective and use cases
- TotalCost journey
- **Paradigm shift: *Classification-by-Property***
- Summary

Problem Analysis

- Algebraic style depends on element sets from system's viewpoints
  - designers constantly revise the optimization model ensuring its consistency with system's engineering model

Problem Analysis

- Algebraic style depends on element sets from system's viewpoints
  - designers constantly revise the optimization model ensuring its consistency with system's engineering model

- *Classification-by-Containment*
  - engineers first identify sets of classifications
  - associate each class with a collection of properties, some being inherent and some reflecting interactions with other classes
  - each class is a containment of individuals having the same characteristics in common
  - each class may then be further specialized into a hierarchy of containments
  - analysis tools use these classifications as first class citizens in any algebraic or analysis model.

New Approach

- Why do we need classes?
  - technical purpose: to help in defining or initializing instances
    - *Classification by Containment* does the job
  - conceptual purpose: to be used for defining effectively common rules or constraints on collection of similar instances
    - *Classification by Containment* fails

New Approach

- Why do we need classes?
  - technical purpose: to help in defining or initializing instances
    - *Classification by Containment* does the job
  - conceptual purpose: to be used for defining effectively common rules or constraints on collection of similar instances
    - *Classification by Containment* fails

- Paradigm shift – *Classification-by-Property*
  - suggested by Parsons and Wand (2000) for information management
  - define *things* that possess properties
    - *intrinsic properties* and *mutual properties*
  - no a-priory classification
  - *classes* are defined by set of properties
    - things could belong to many classes
  - *instance, class* and *property bases*

Classification-by-Property API

- **Scope**(A), where A is a set of attributes
  – returns instances that have all attributes in set A
  – e.g., **Scope**({Cost}) returns all instances that have attribute "Cost"

DSE Ontology

– Attributes
  - variables
  - parameters

– ***isSelected***

– ***isContainedIn(instance)***

– ***isMapped***

– ***couldBeMapped(instance)***

totalCost journey – Magic of the New Approach

$$totalCost(i)$$

$$= \sum_{j \in \mathbf{Scope}(\{\text{Cost, isContainedIn}(i)\})} isSelected(j) \cdot Cost(j) \quad \forall\, i \in \mathbf{Scope}(\text{totalCost})$$

// Mapping

$$\sum_{j \in \mathbf{Scope}(\text{couldBeMapped}(i))} isMapped(i)(j) = isSelected(i)$$

$$\forall\, i \in \mathbf{Scope}(\text{isRequirementsElement})$$

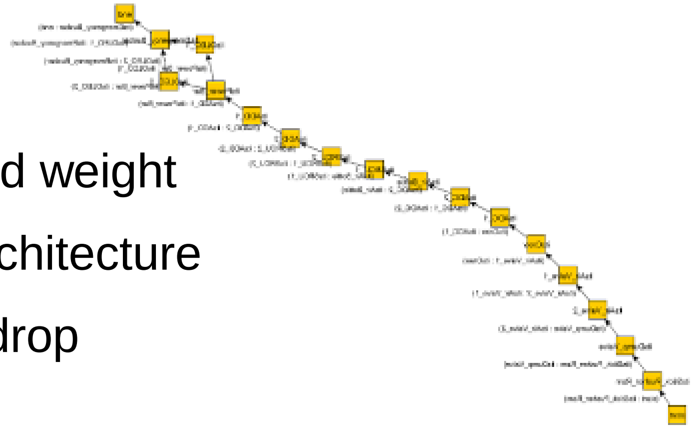
totalCost journey – Magic of the New Approach

- Resource viewpoint
  - capacity of architectural component *j* for resource *i*, $\mathrm{dseCapacity}(i)(j)$, is not exceeded by granting all the requests, $dseRequest(i)(k)$, from requirements mapped to it

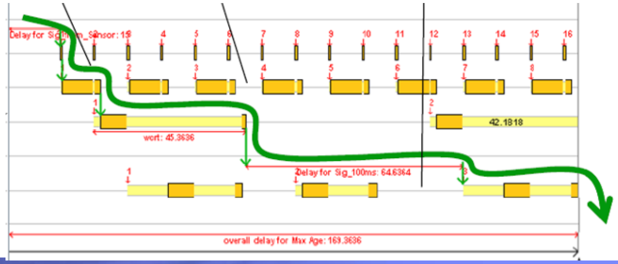
$$dseCapacity(i)(j) \geq \sum_{k \in Scope(dseRequest(i))} isMapped(k)(j) \cdot dseRequest(i)(k)$$

$$\forall\, i \in dseResources, j \in Scope(dseCapacity(i))$$

Current Metrics / Analysis Patterns



- Minimizing overall Architecture Cost
- Minimizing overall weight, and wiring length and weight
- Mapping functional requirements to physical architecture
- Optimize Power distribution, Minimize voltage drop
- Data flow
- Optimize geographical coverage and positioning
- Optimal placement of assets, sensors and ground fields
- Scenarios fulfillment
- Reliability: Approximate algebra to take Reliability into account for optimization
- Timing: Approximate algebra to take Timing into account for optimization
- Interface compatibility
- Resource allocation



Agenda

- Approaches to Architectural Design
- Background: Architectural Optimization Workbench (AOW)
- Reusable system analysis: objective and use cases
- TotalCost journey
- Paradigm shift: *Classification-by-Property*
- **Summary**

Main Findings

- Using tools for DSE and analysis requires reusable analysis libraries
  – similar to component libraries in modeling tools

- In system design, several analysis metrics/patterns are repeatedly used
  – mapping functionality to architecture, reliability, resource allocation, energy and data flow

- Current analysis modeling does not support building reusable analysis libraries)

Main Recommendations

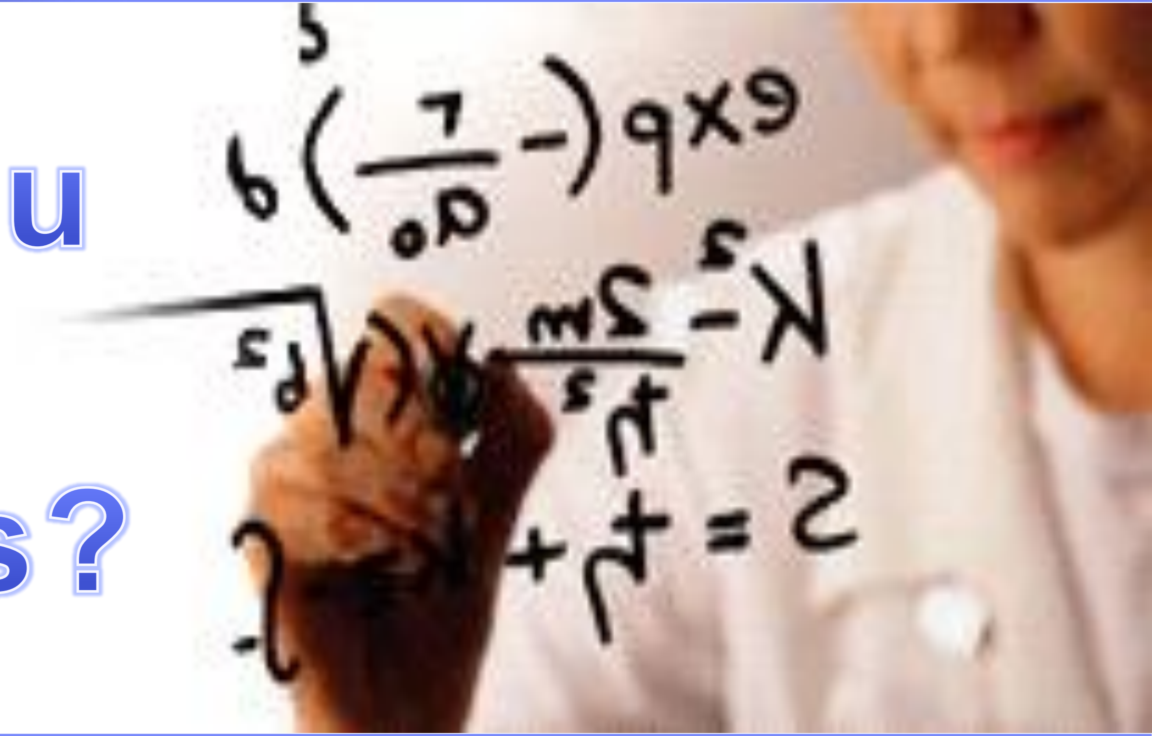
- The main problem with the existing methodology is the definition of sets – *Classification- by-Containment*
  - good to define concepts and initialize values
  - fails to define sets needed for analysis
- New paradigm of *Classification-by-Properties* defines robust sets for analysis
- Ontology based concepts and property-defined sets enable building reusable libraries for system optimization and analysis
- Current reusable libraries
  - mapping, reliability, resource allocation, interface compatibility, power distribution, voltage drop, data flow, … (ongoing extension)

Future directions

- Extending the proposed approach to dynamic systems
  - reusable state charts / activity diagrams
- Optimization patterns in building Analysis Viewpoints constraints
  - symmetry breaking

Thank you

Questions?



Where can we use this approach?

- Safety Critical Systems
  - Systems with redundant functions (flight control, hydraulic systems, Automotive safety systems etc.)
- Network systems (data, power, energy, fluid)
  - Integrated Modular Avionics, AutoSAR
  - Electrical Architecture (e.g. GM global-B)
  - Cabin Intercommunication system
  - Entertainment systems
  - Fuel and cooling systems
  - Electric power systems
- Complex multi-physics systems
  - For example any mix of the above with an electronic/software control
- Cabin Architecture
- System of Systems
  - Scenarios handling