Generic Approach to Architecture Optimization

IBM Research - Haifa

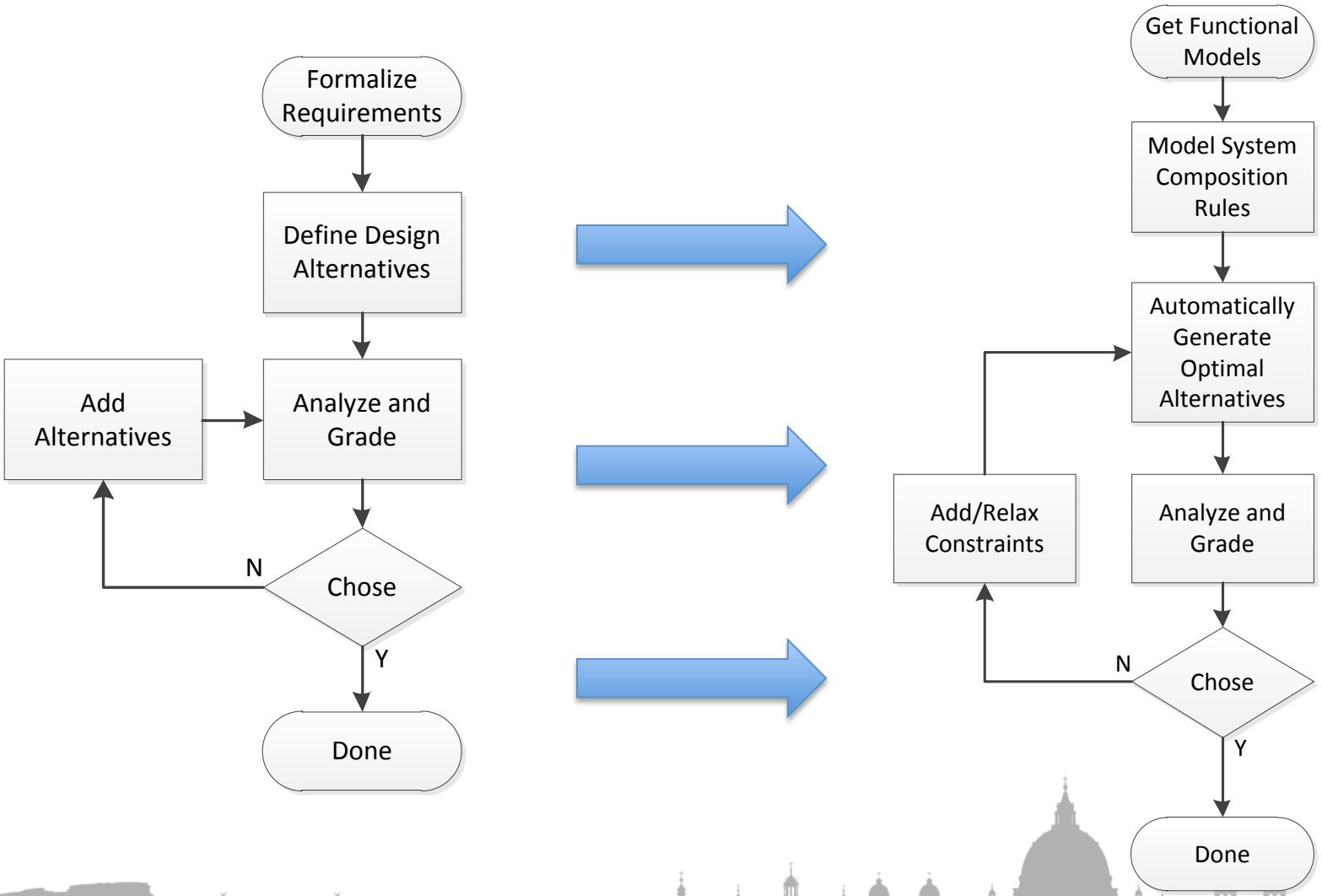
Henry Broodney, Lev Greenberg, Michael Masin, Dolev Dotan

Architectural Design - Problems and Remedies

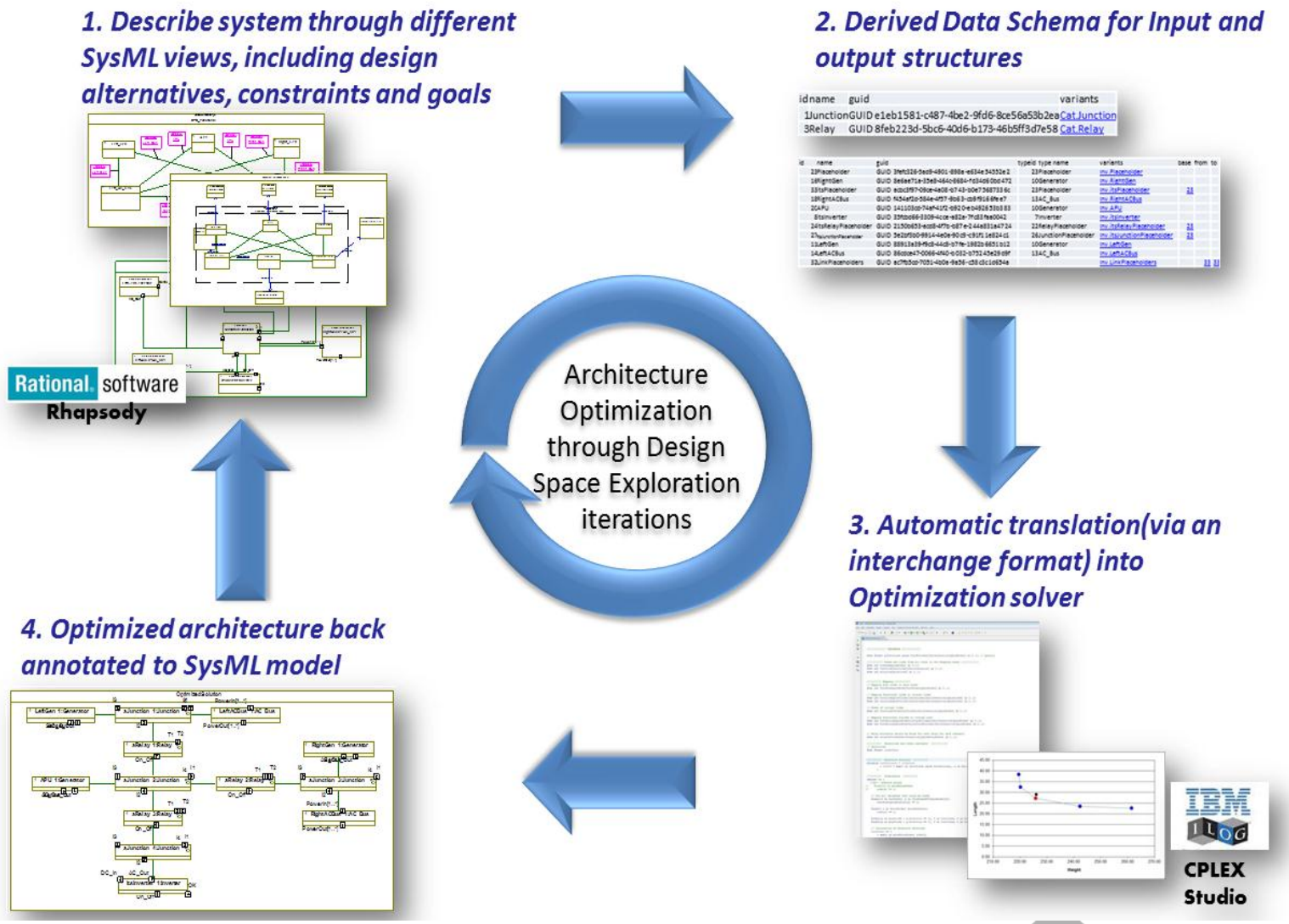
- Key concerns:
  - System Complexity is increasing – manual decisions no longer possible
  - Design Space Exploration tools today – Excel and Visio
  - Optimization requires a math expert to work with the Systems Engineer
    - Lots of information is lost in the interaction
  - Tedious and manual management of such processes

- Solution approach:
  - Allow the Systems Engineer to formally express the system composition rules and constraints
  - Automatically generate the optimization code from the model
  - Visualize results
  - Management tool orchestration and collaboration

Architectural design flow – now and next



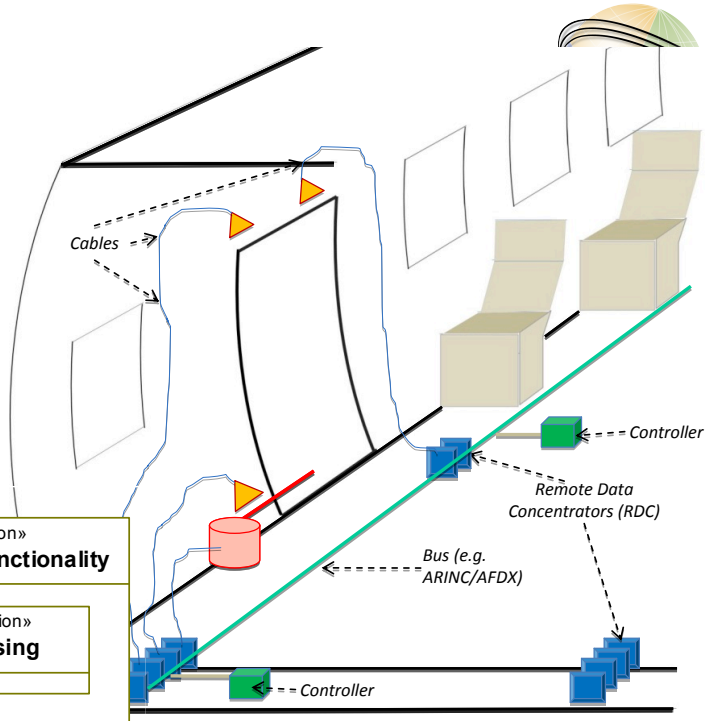
Architecture Optimization Workbench Concept

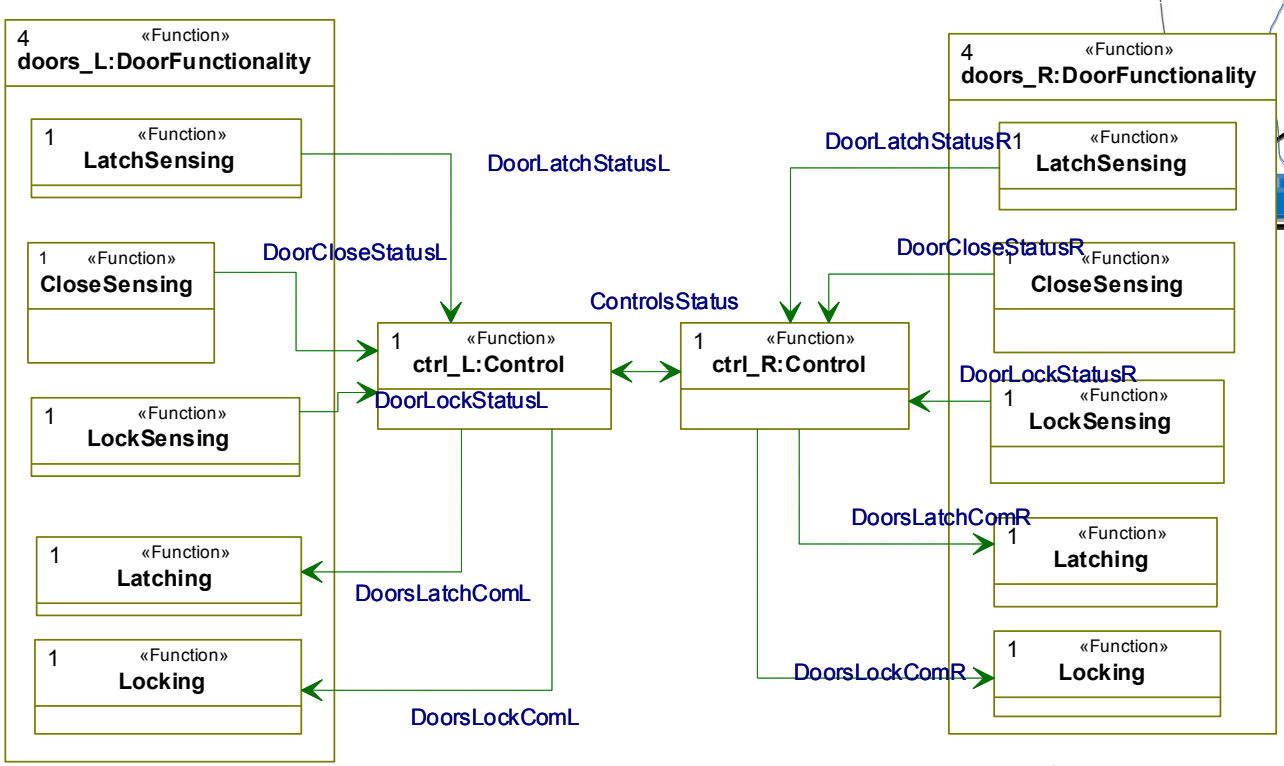


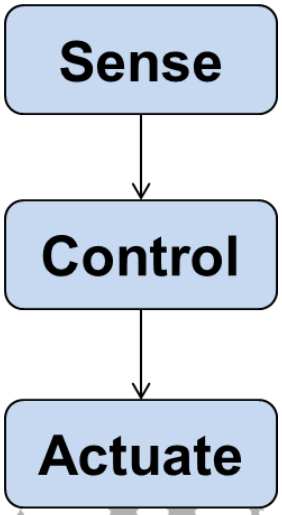
* This work was partially sponsored by DARPA under contract #FA8650-10-C-7074.

From user story to model

Functional Model

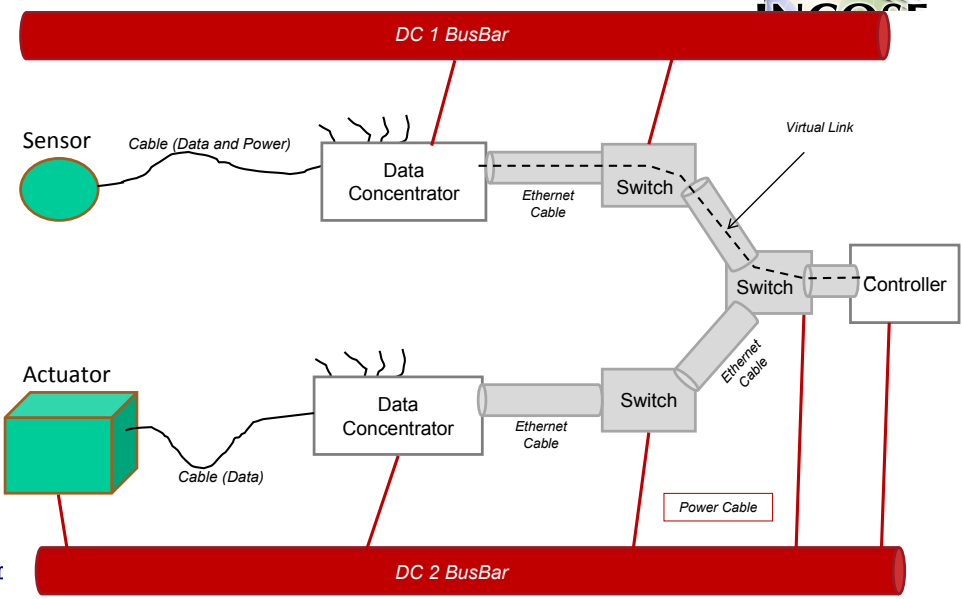


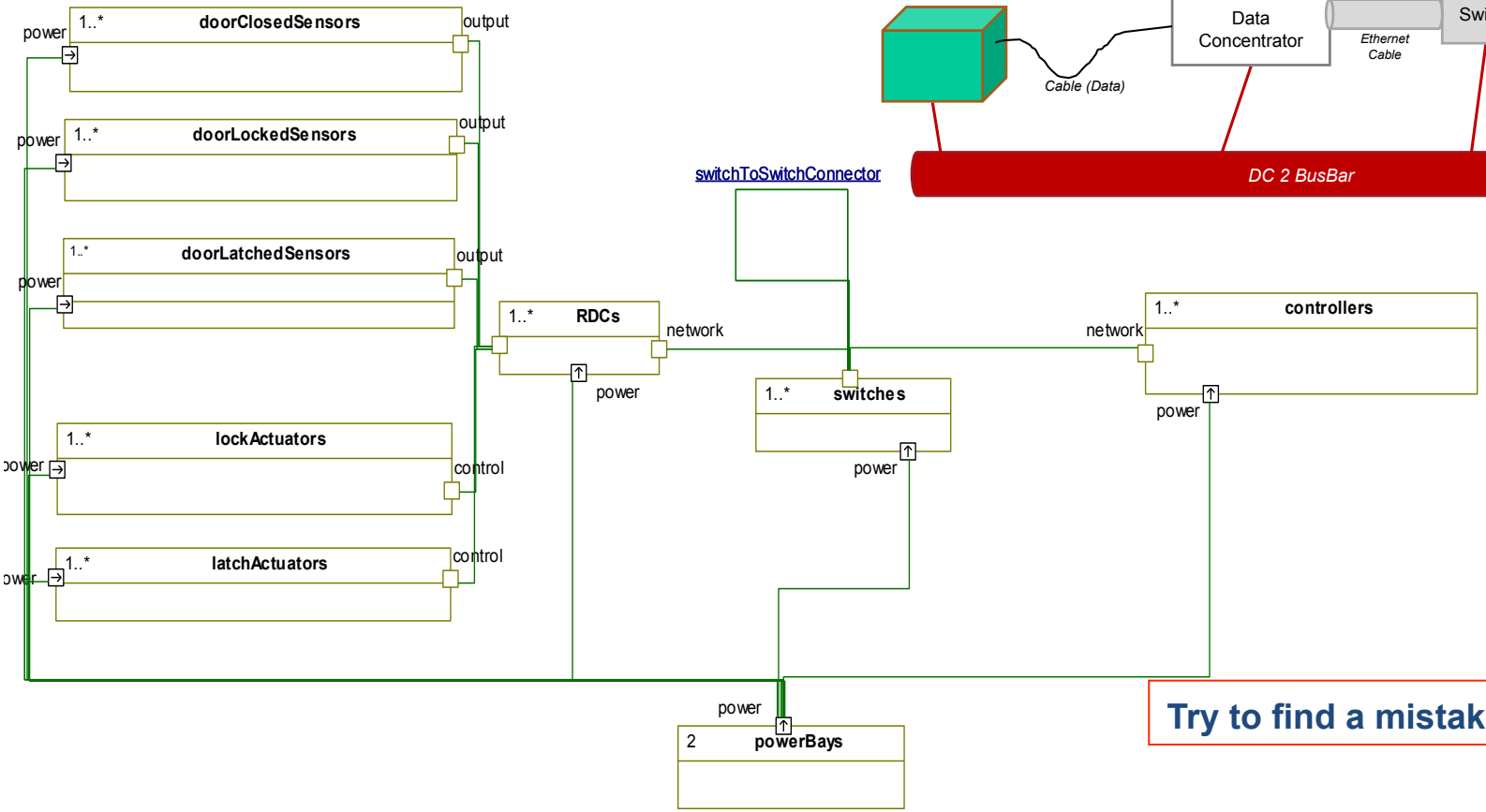




From user story to model

Architectural Template

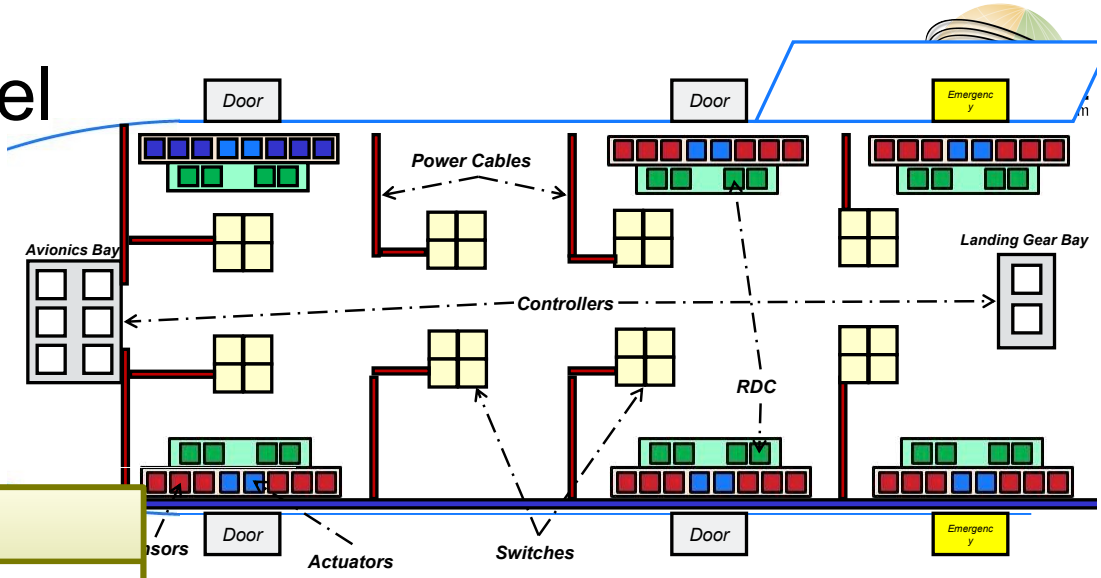


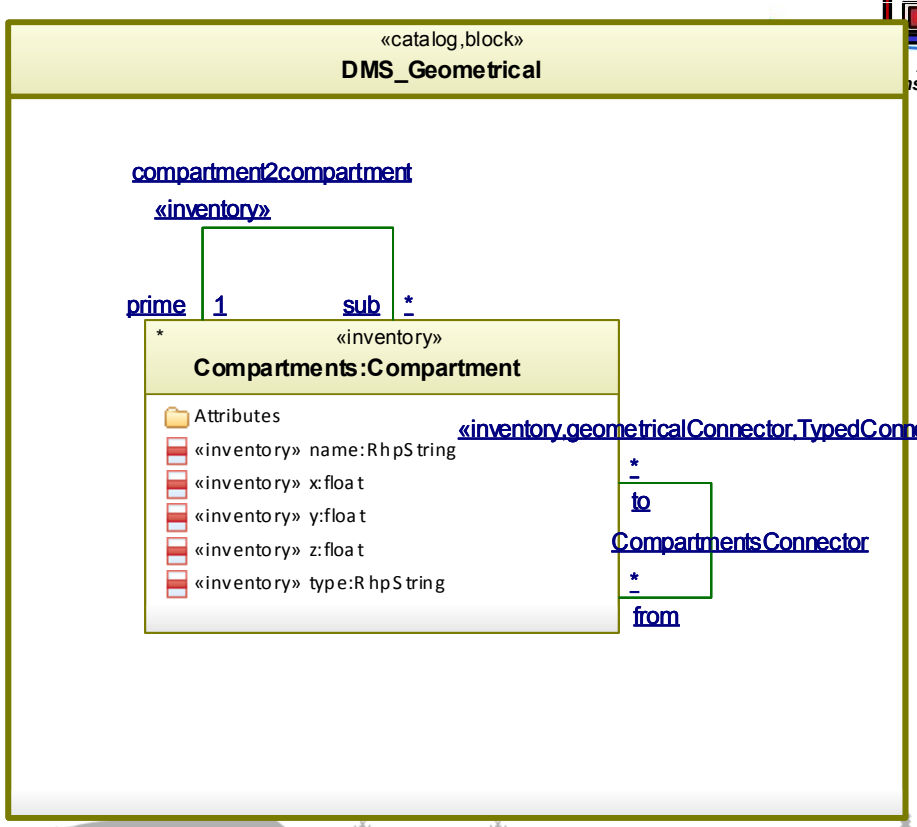


Try to find a mistake in the model

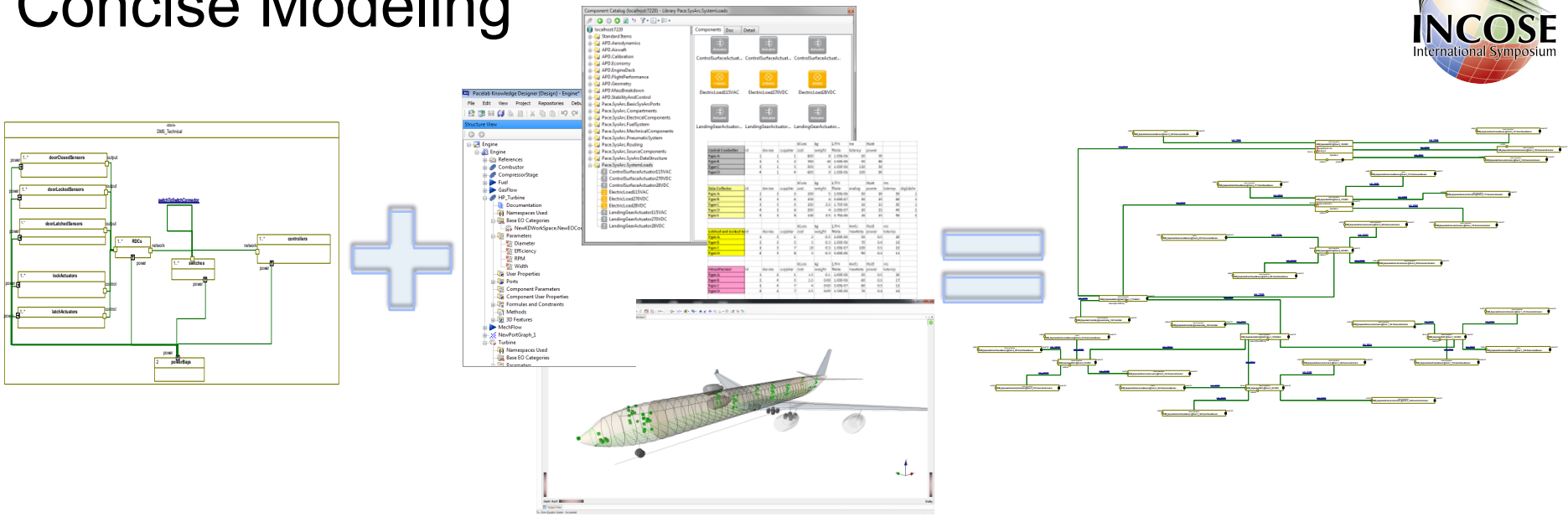
From user story to model

Geometrical Model

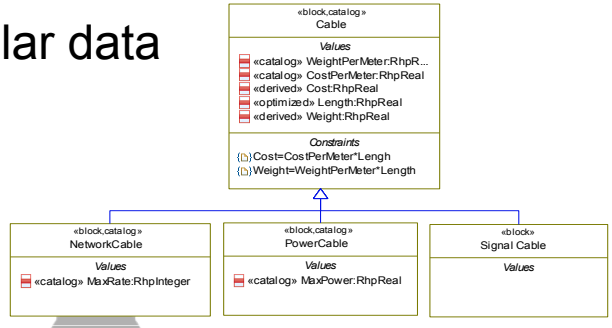




Concise Modeling

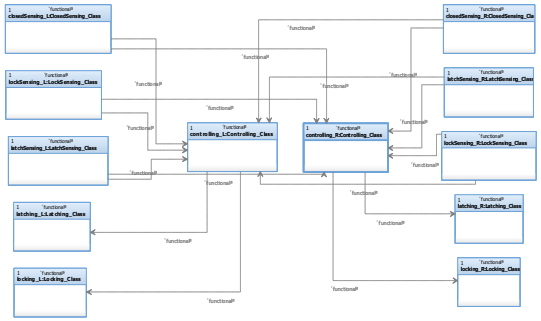


- Large systems architectures are difficult to model
  - Lots of elements and details
  - Time consuming
  - Error prone
- Concise Modeling – SysML models combined with tabular data
  - SysML depicts the system composition rules
  - Tables contain instantiations
- For an existing model the tabular data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

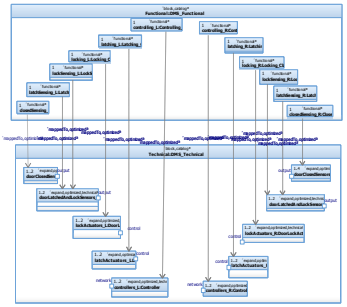


Concise modeling: views and relations

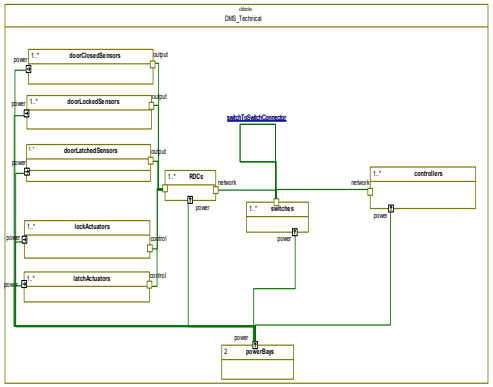
Functional



Mapping

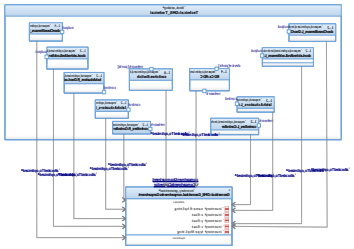


Technical

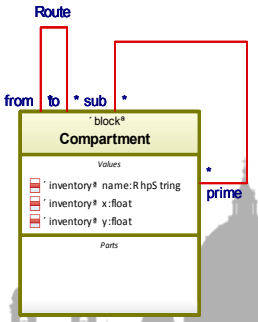


- Mapping is a means of defining what goes where
- Functions are mapped to a type of component or to a specific instance
- Components are mapped to a type of compartment or to a specific one

Mapping

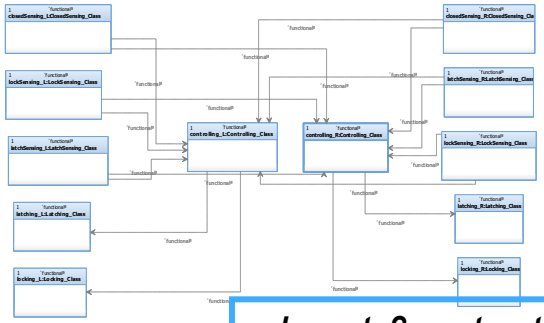


Geometrical

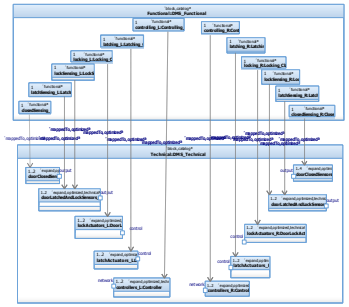


Concise modeling: views and relations

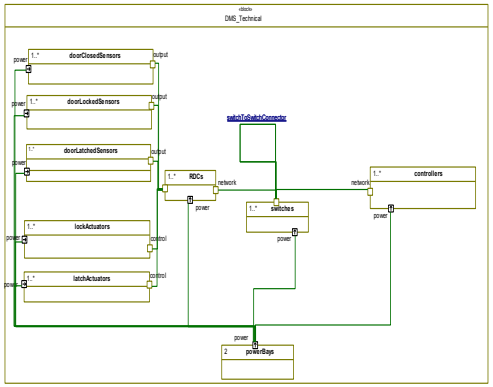
Functional



Mapping



Technical



Automatic Generation

Input & output structures

int	int	int
*id	sub	prime

Relations

int	int	string	int	float	float	float	float	int	float
*id	*typeid	typeName	ports	cost	fRate	latency	power	supplier	weigl
1	0								
2	0								
3	0								
4	0								
5	0								
6	0								

Inventory

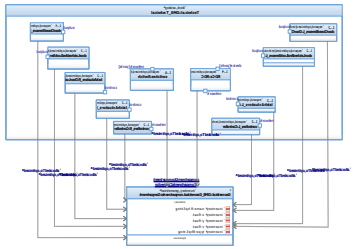
int	int	int	int	float	float	float
*id	*typeid	area	device	x	y	z
1	37					
2	37					
3	37					
4	37					
5	37					
6	37					

Indices

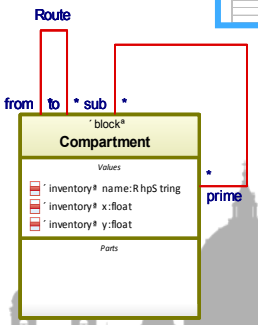
PlaceId	DeviceId		linkType	cableId	linkId	end1	end2	Cost	Weight	Length

Output Tables

Mapping

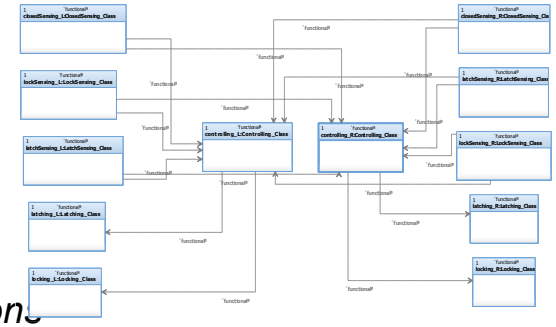


Geometrical

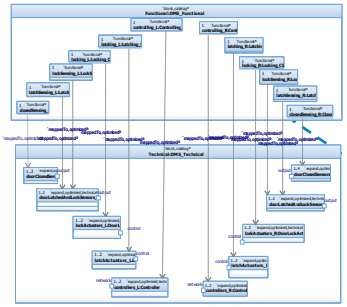


Concise modeling: views and relations

Functional



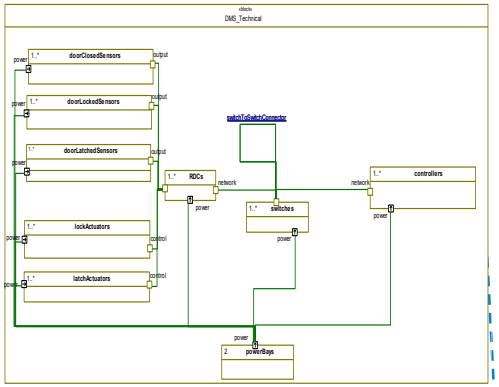
Mapping



Relations

int	int
id	base
1	105
2	106
3	107
4	108
5	109
6	110
7	111
8	112
9	113
10	114

Technical

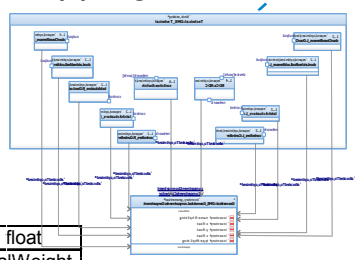


Geometry

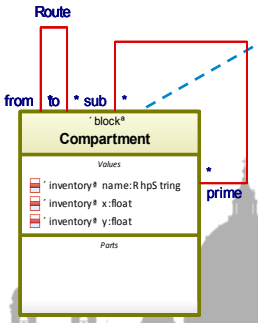
int	float	float	float
id	x	y	z
1	2.2	0.5	-1
2	2.7	0.5	-1
3	2.2	0.3	-1
4	2.7	0.3	-1
5	2.2	0.1	-1
6	2.7	0.1	-1
7	2.2	-0.1	-1
8	2.7	-0.1	-1
9	2.2	-0.3	-1
10	2.7	-0.3	-1
11	2.2	-0.5	-1
12	2.7	-0.5	-1

Component Library (PDM)

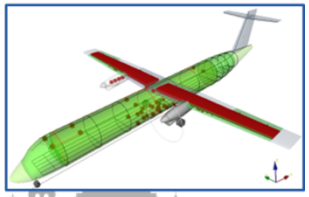
Mapping



Geometrical



Data

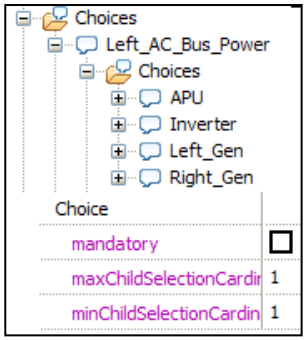


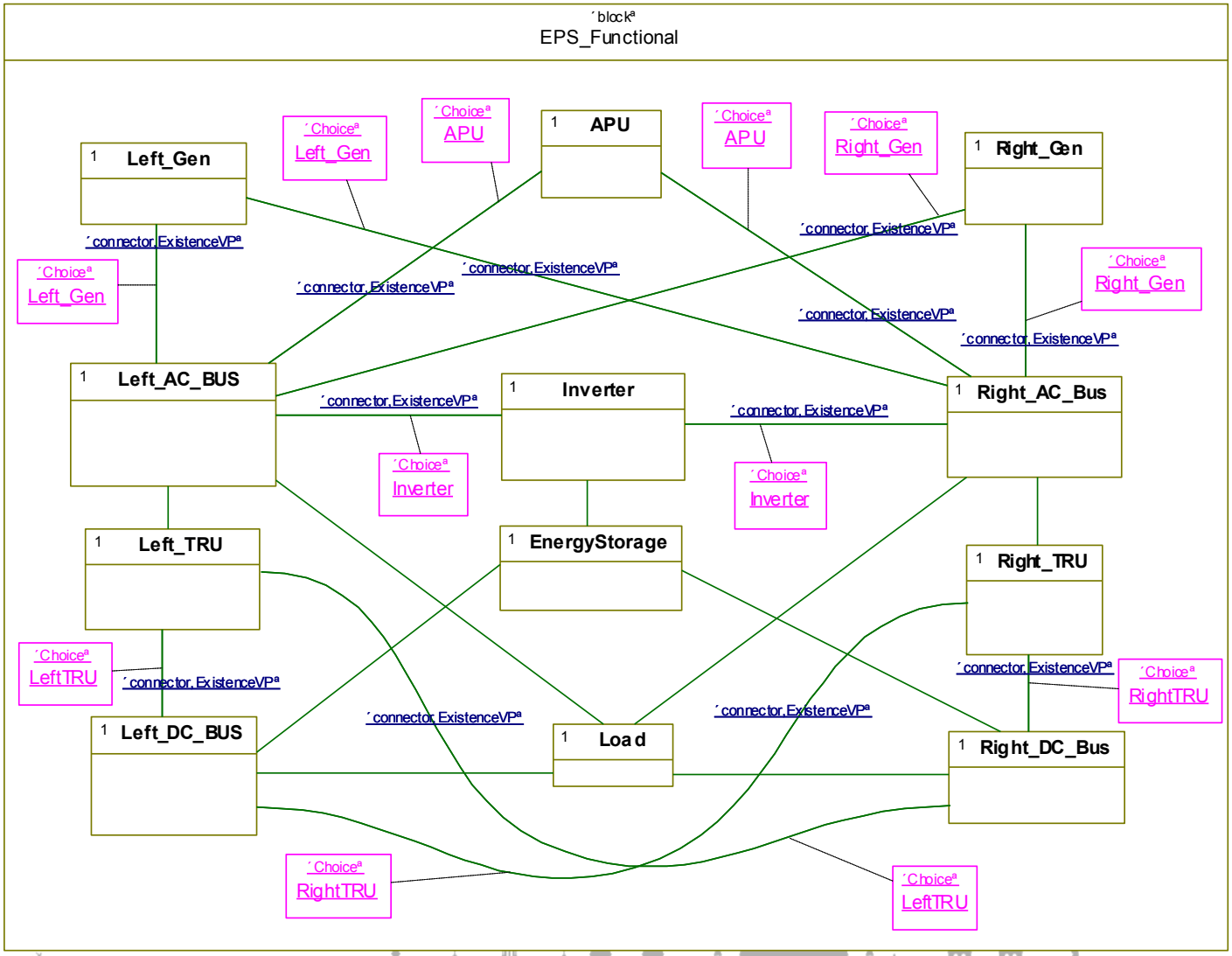
Catalog

int	float	float	float	int	int	float	int	float
id	fRate	latency	power	analog	cableType	totalCost	supplier	totalWeight
1	2E-06	50	30	50	1	100	3	5
2	5E-07	40	25	30	1	150	4	4
3	2E-06	50	15	16	1	100	5	2.5
4	5E-07	40	25	30	2	200	6	4
5	2E-06	50	15	16	2	130	6	2.5

Modeling behavioral constraints

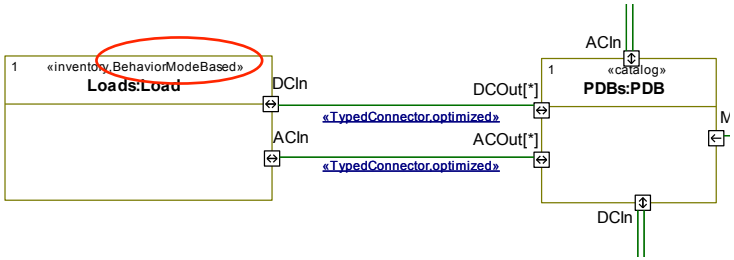
- Use variability constructs in functional view
- Add system operational scenarios
  - Extended OCL
  - Attributes as arguments

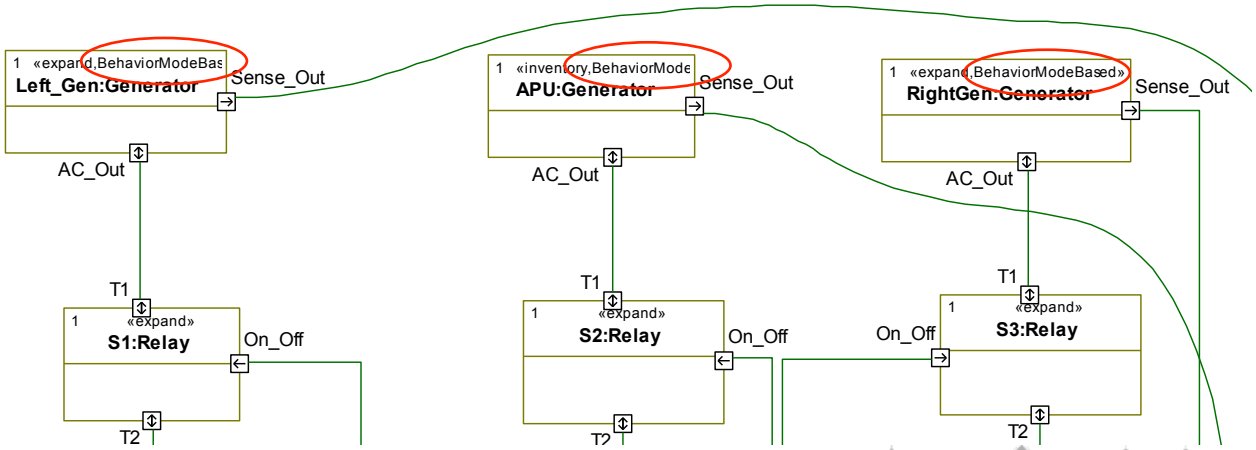




Scenario Modeling

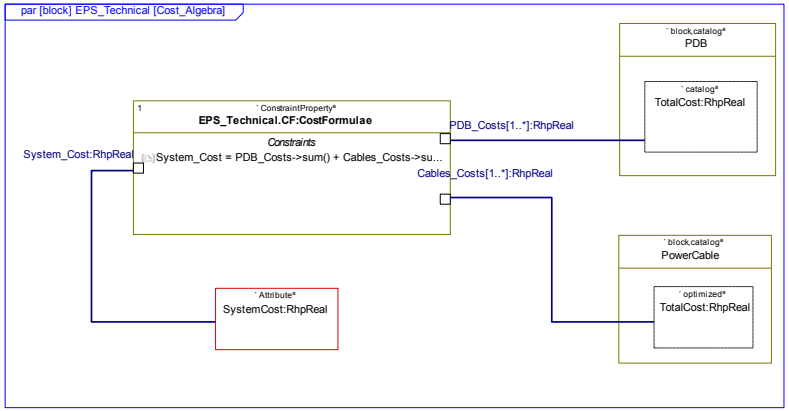
- Systems have many operating modes and scenarios
  - Design must satisfy all
  - Modeling tools are needed to specify modes and scenarios
- Variability package extension
  - SysML based
  - Behavior modes groups
  - Modeling operating and failure modes





Objectives and constraints modeling - Algebras

- Use SysML parametric diagram or textual notation
- Some metrics will be only referenced due to their complexity
- Use the object model (including concise) to define sets
- Model interrogation functions – additional means to define sets in absence of the actual architecture
- Stereotypes with tags to select objectives and define their constraints



Parametric Diagrams

«block»
EPS

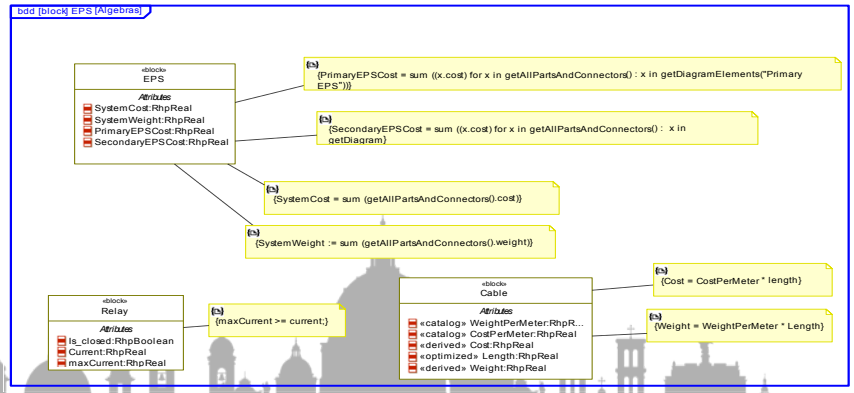
Values

«objective» NumberOfRelays:int

Constraints

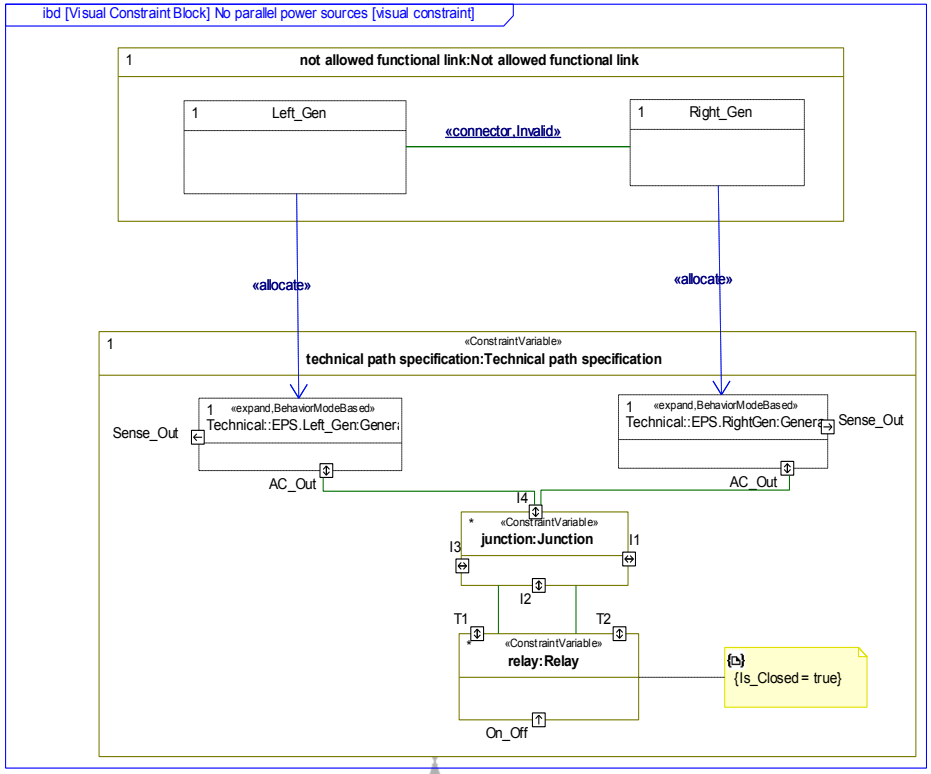
{} NumberOfRelays=count(getAllParts(Relay))

Stereotyped Comments



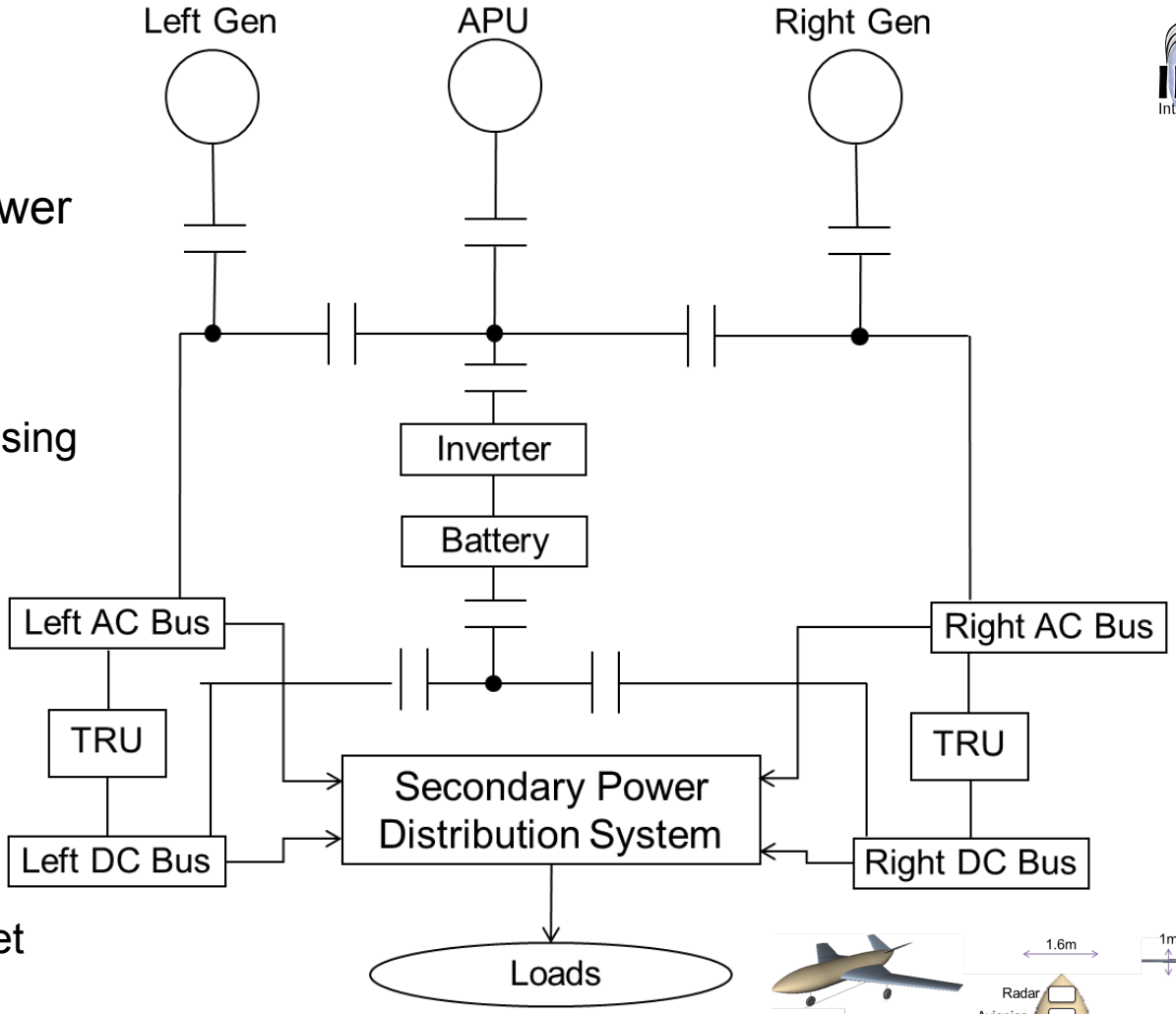
Constrains modeling - structure

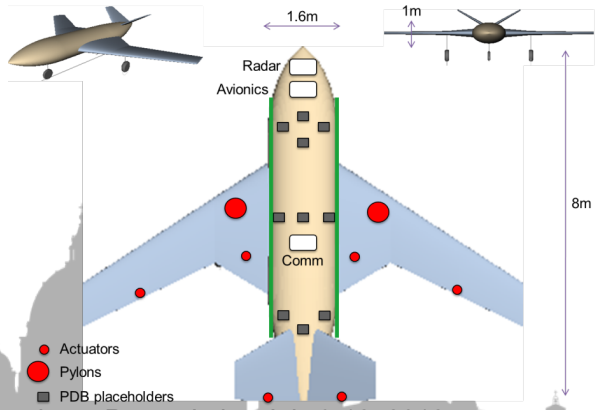
- Constraints are given on the functional (requirements) level
  - Mapping needed to technical level
- New concept – constraints diagram
  - Model a functional link
  - Map to a structural template
  - Add link modifier
    - Should/should not exist
    - Related operational scenario
    - Probability information
    - Reliability/Timing/etc constraints on link



DSE Use case

- EPS – Electrical Power System for UAV
  - Three generators
  - Single controller managing the sensing and switching
  - Loads on the secondary power system
- Objective
  - Optimize for cost and weight
  - Ensure system constraints are met

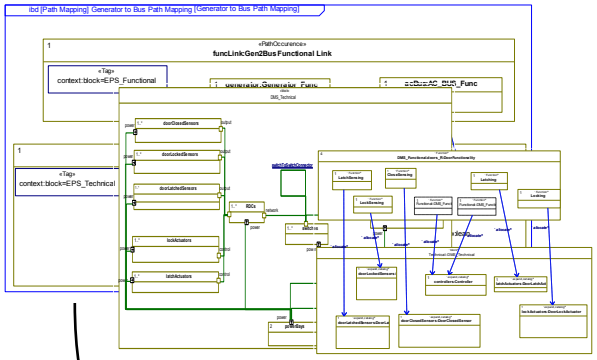




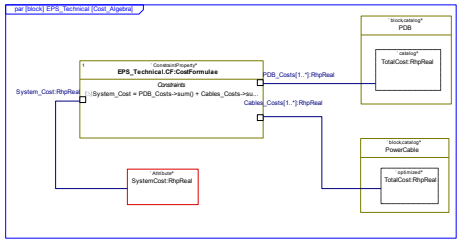
* Courtesy of United Technologies Corporation

Optimization Auto-Generation

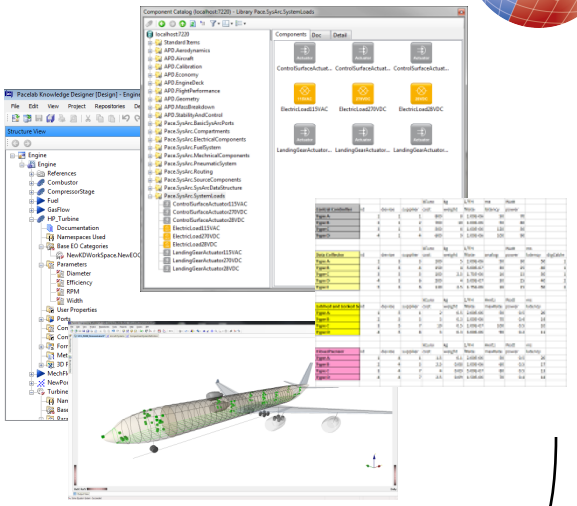
Requirements & Constraints



Metrics



Component library



Automatic Generation

Optimization Problem Formulation (CPLEX)

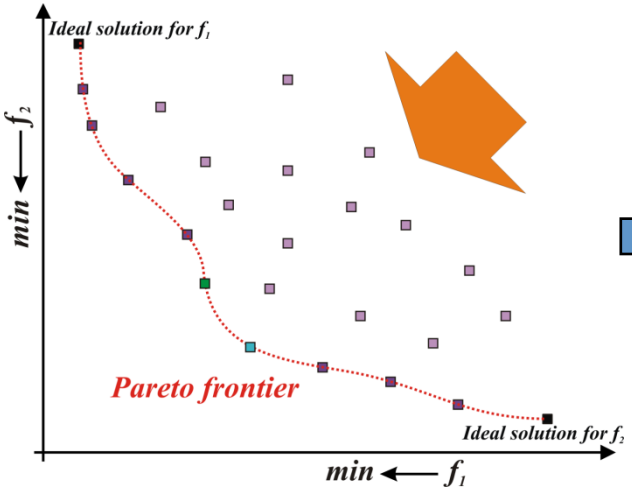
```
minimize weightPriority * totalWeight + lengthPriority * totalLength;

//////////////////// 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];
```

Multi Objective Optimization

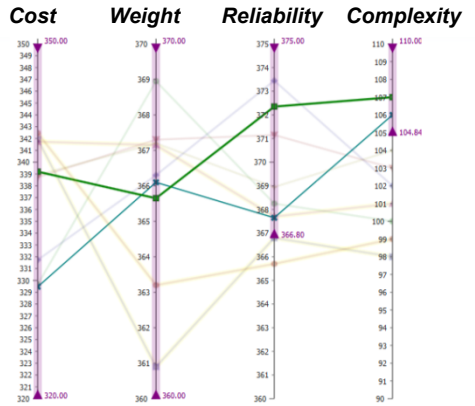
Optimization Problem Formulation (CPLEX)

```
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
//
//  UsePAConst:
//      forall(j in places : j.id == 1) usePlace[j] == 1;

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

  TotalWeightConst:
    totalWeight == sum(p in places, p in pdbs) pdb4place[p][p] * 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)
  UptoOnePDBtoPlaceConst:
    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];
```

Calculation of Efficient Frontier

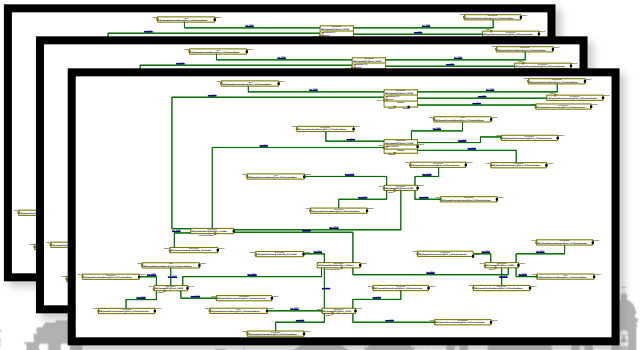


Multi Objective visualization



Selection of desired options

Architectures' visualization

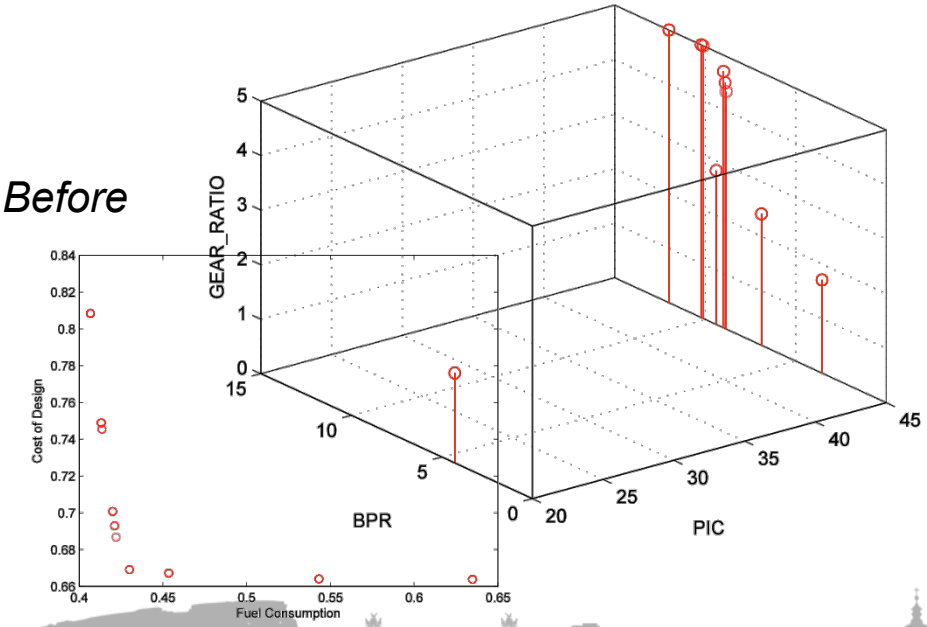


Next iteration

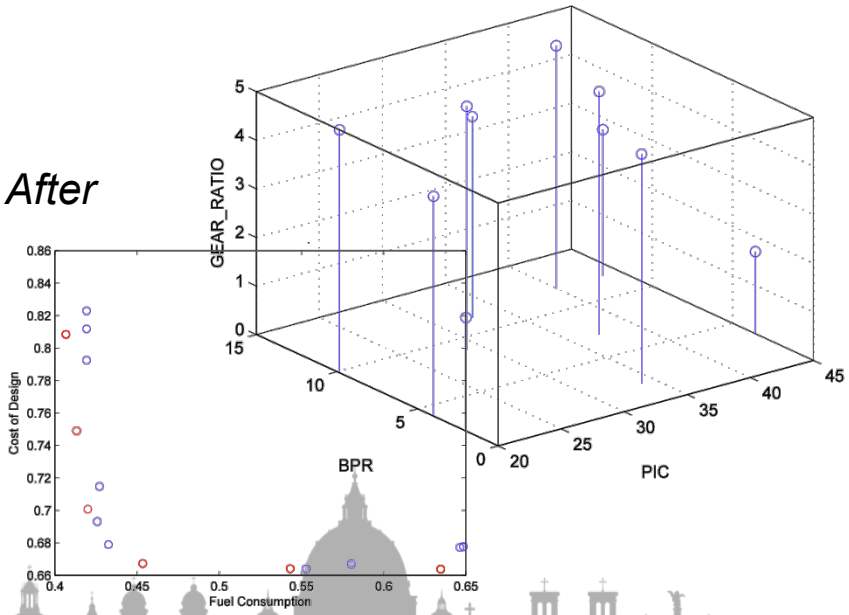
Diversity Maximization Algorithms

- Points on efficient frontier often represent very similar architectures
  - No real choice is given
  - No boost in innovation
- Existing genetic algorithms do not guarantee maximum diversity and are computationally heavy.
- Our algorithms guarantee maximal designs diversity under allowed deviation (distance) from the efficient frontier

Before

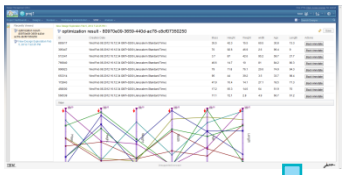
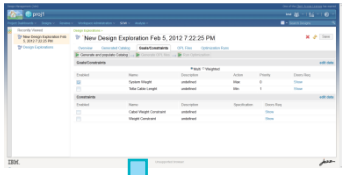
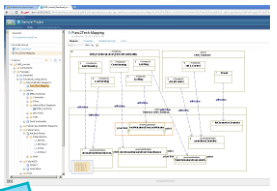


After



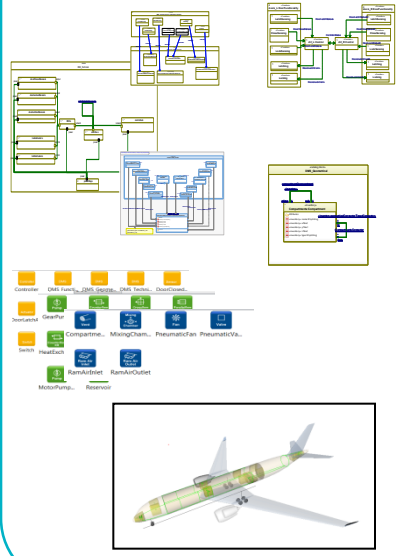
Process and Tools overview

Architect's cockpit

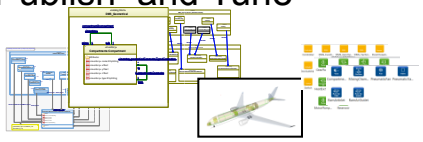
Requirements

Conceptual design With Concise Modeling & Component Libs



Perform Design Exploration

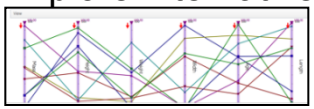
Publish and Tune



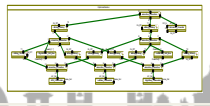
Auto Translate to Optimization Model & Solve

ILOG CPLEX Faster and Smarter Than Ever

Explore Alternatives



Visualize Alternatives



Analysis & Simulation

R

DESYRE

INCHRON THINK REAL-TIME

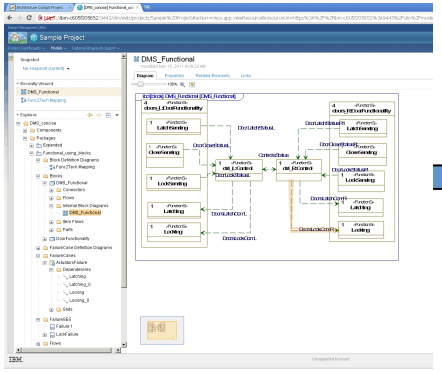
ANSYS CFX

PACE

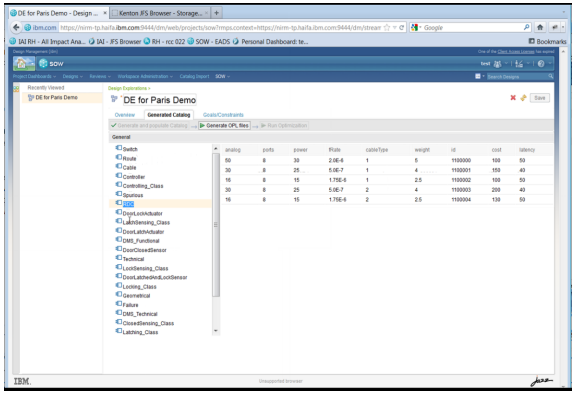
PSpice®

Architect Cockpit

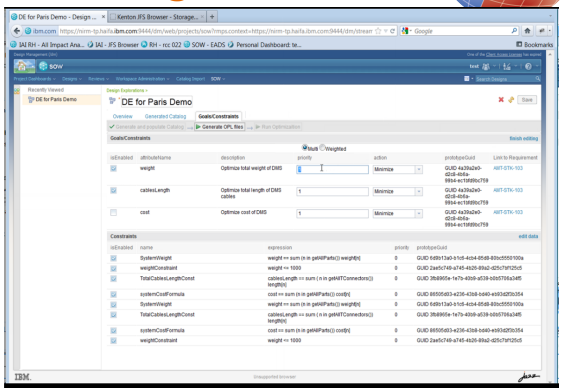
Modeling management



Data management

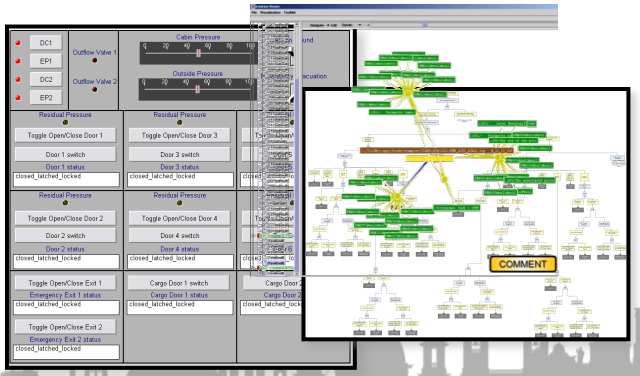


Objectives management

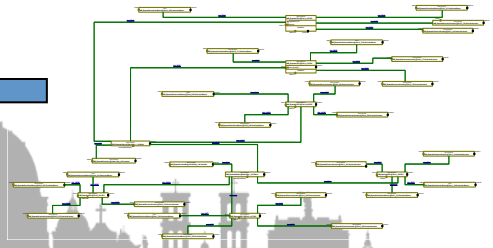


Results management

Multiple Analyses



Back-Annotation



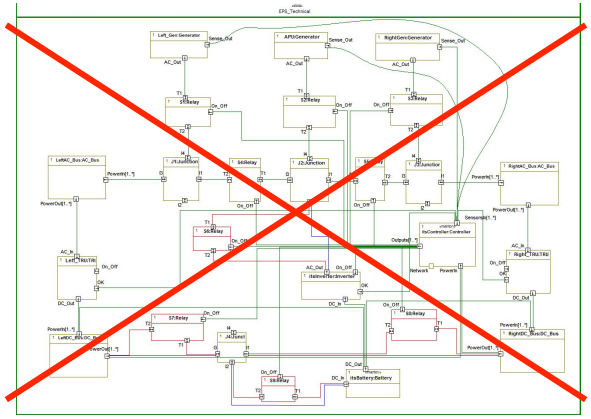
- Single environment
  - Main tool of the Systems Engineer
  - Controlling the design and analysis process
- Based on Design Manager and JTS
- Interaction with modeling environments
  - Review and comment mechanisms
  - Cross organizational collaboration
  - Models import / export control
  - Back-end model transformations
- Integration with analysis tools
  - Simulations, computations, domain specific views
  - White-box, black-box
  - Analysis results feedback into models
- Visual analytics

Thank you

Questions?

Design Space Exploration - METHODS

- Define SysML extensions for concise systems modeling
  - Composition rules rather than complete structures
  - Add structural constraints
  - Component libraries as third party input via structured interchange format
  - Define contracts for the system and its components – behavioral constraints
  - Define variables algebras (e.g. weight, cost, complexity etc)
- Add variability points - choices
- Add optimization enabling data to the model
  - Define objectives
  - Add constraints



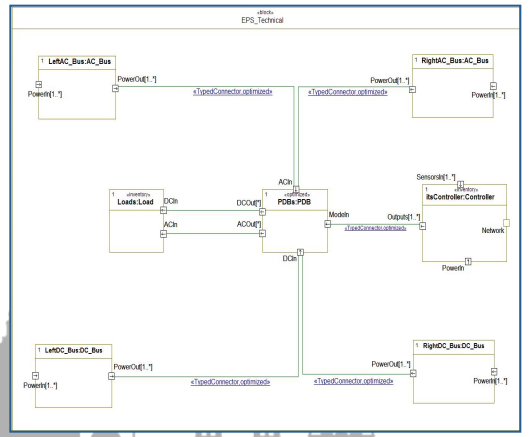
GCU.C1.Assumption:= [**E** *"Gen.failed"*] **implies** [**C** *Gen.status=failed]* **holds forever**

GCU.C1.Guarantee:=whenever [**E** "*GCU.recv_gen_failure*"] **occurs** [**E** "*GCU.send_gen_failure*"] **within** (+*GCU_send_delay*)]

GCU.C2.Guarantee:=whenever [**E** "*GCU.recv_gen_failure*"] **occurs** [**E** "*GCU.send_open_gen_relay*"] **within** (+*GCU_send_delay*)]

GCU.C3.Guarantee:=whenever [**E** "*GCU.recv_command* com" **and not (** com=="*send_open_gen_relay*" **and C** *GCU.gen_status_failed*)] **occurs** [**E** "GCU.send_command com"] **within** (+*GCU_send_delay*)]

+



+

int	int	string	int	float	float	float	float	int	float	int	int
*id	*typeid	typeName	ports	cost	fRate	latency	power	supplier	weight	vls	cableType
1	0										
2	0										
3	0										
4	0										
5	0										
6	0										

PlaceId	DeviceId	linkType	cableId	linkId	end1	end2	Cost	Weight	Length

int	int	int
*id	sub	prime