Massachusetts Institute of Technology

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

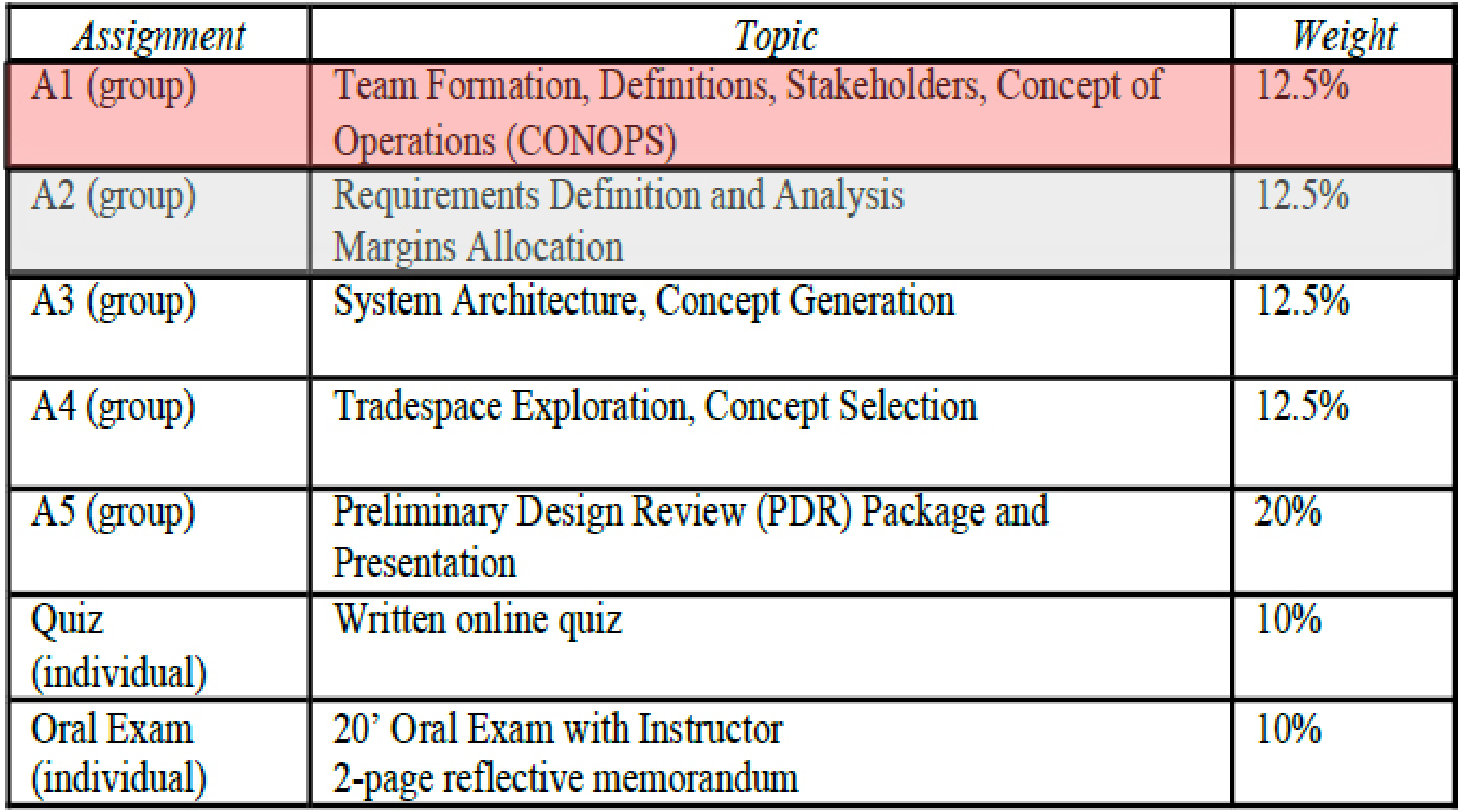
Fundamentals of Systems Engineering

Prof. Olivier L. de Weck, Mark Chodas, Narek Shougarian

Session 3

System Modeling Languages

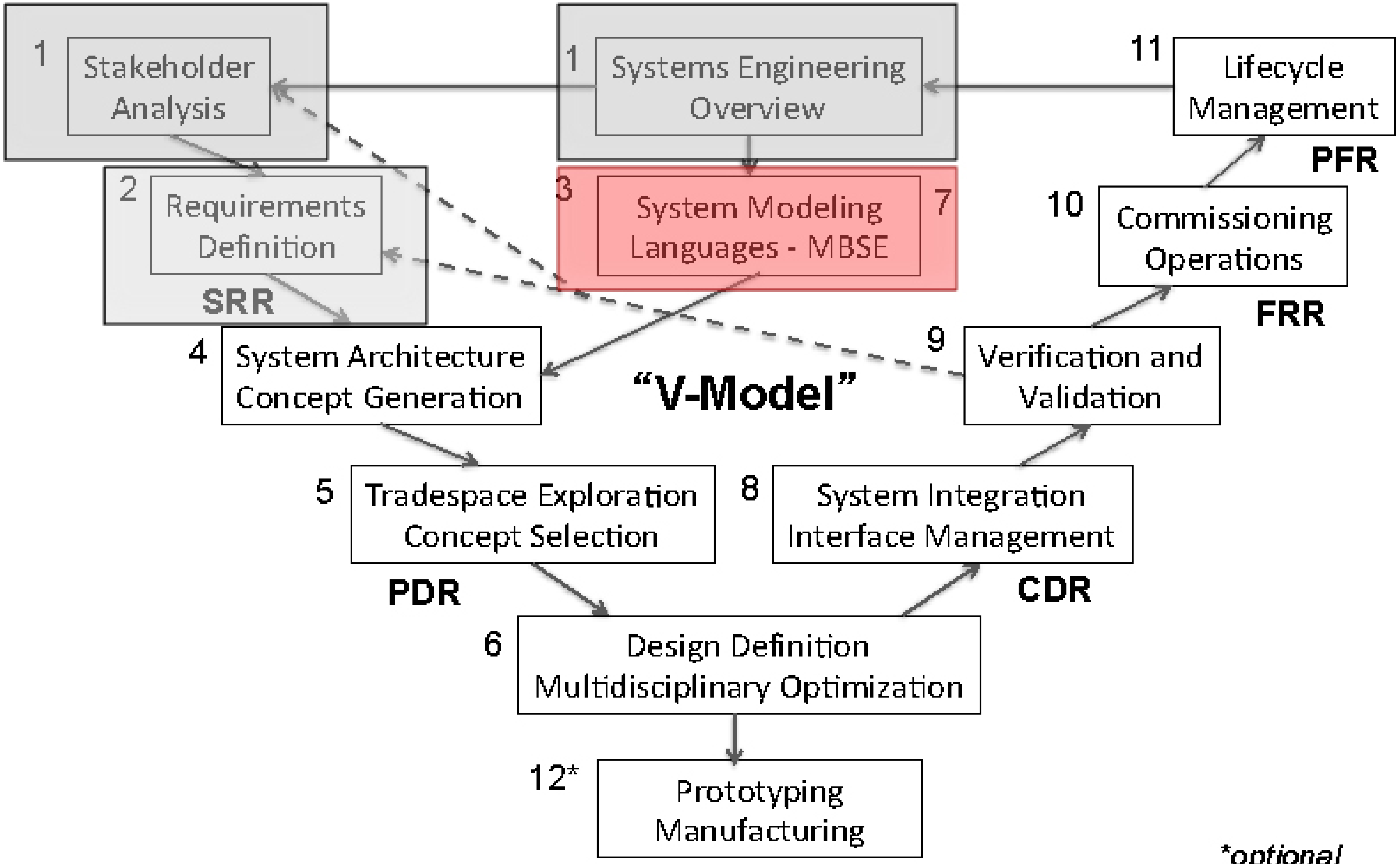
Reminder: A1 is due today !



Assignment	*Topic*	*Weight*
A1 (group)	Team Formation, Definitions, Stakeholders, Concept of Operations (CONOPS)	12.5%
A2 (group)	Requirements Definition and Analysis Margins Allocation	12.5%
A3 (group)	System Architecture, Concept Generation	12.5%
A4 (group)	Tradespace Exploration, Concept Selection	12.5%
A5 (group)	Preliminary Design Review (PDR) Package and Presentation	20%
Quiz (individual)	Written online quiz	10%
Oral Exam (individual)	20' Oral Exam with Instructor 2-page reflective memorandum	10%

The “V-Model” of Systems Engineering

16.842/ENG-421 Fundamentals of Systems Engineering


1
Stakeholder Analysis
1
Systems Engineering Overview
11
Lifecycle Management
PFR
2
Requirements Definition
SRR
3
System Modeling Languages - MBSE
7
10
Commissioning Operations
FRR
4
System Architecture Concept Generation
“V-Model”
9
Verification and Validation
5
Tradespace Exploration Concept Selection
PDR
8
System Integration Interface Management
CDR
6
Design Definition Multidisciplinary Optimization
12*
Prototyping Manufacturing

*optional

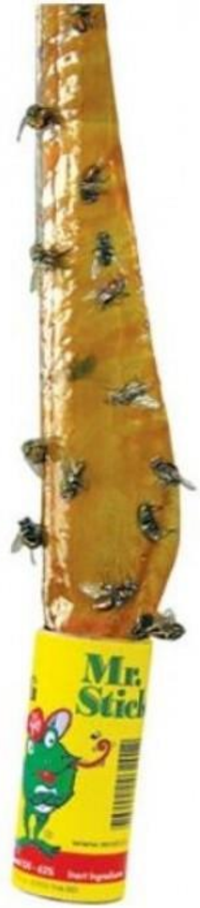
Numbers indicate the session # in this class

Overview

- Why Systems Modeling Languages?
    - Ontology, Semantics and Syntax
- OPM – Object Process Methodology
- SySML – Systems Modeling Language
- Modelica
- What does it mean for Systems Engineering of today and tomorrow (MBSE)?

Exercise: Describe the "Mr. Sticky" System

- Work with a partner (5 min)
- Use your webex notepad/white board
- I will call on you randomly
- We will compare across student teams



Why Systems Modeling Languages?

- Means for describing artifacts are traditionally as follows:
  - Natural Language (English, French etc.…)
  - Graphical (Sketches and Drawings)
  - These then typically get aggregated in "***documents***"
    - Examples: Requirements Document, Drawing Package → Technical Data Package (**TDP**) should contain all info needed to build and operate system

- **Advantages** of allowing an arbitrary description:
  - Familiarity to creator of description
  - Not-confining, promotes creativity

- **Disadvantages** of allowing an arbitrary description:
  - Room for ambiguous interpretations and errors
  - Difficult to update if there are changes
  - Handoffs between SE lifecycle phases are discontinuous
  - Uneven level of abstraction
  - Large volume of information that exceeds human cognitive bandwidth
  - Etc.…

System Modeling Languages

- Past efforts to create System Modeling Languages
  - E.g. Bond Graphs (1960), IDEF (1981), etc…
- Regardless of the System Modeling Language being developed and used they share the common features:
- Domain agnostic
- **Ontology https://en.wikipedia.org/wiki/Ontology_engineering**
  - Defines the entities that are allowed to exist and be described
  - How these entities can be grouped, related to a hierarchy and subdivided
  - Constrains the universe of concepts that can be represented in the language
- **Semantics https://en.wikipedia.org/wiki/Semantics**
  - Describes the meaning of the entities allowed by the ontology
  - Relationship between signifiers (e.g. words, symbols …) and their denotation
- **Syntax https://en.wikipedia.org/wiki/Syntax**
  - Set of rules, principles and processes that govern the structure of the language and how correct "sentences" can be synthesized

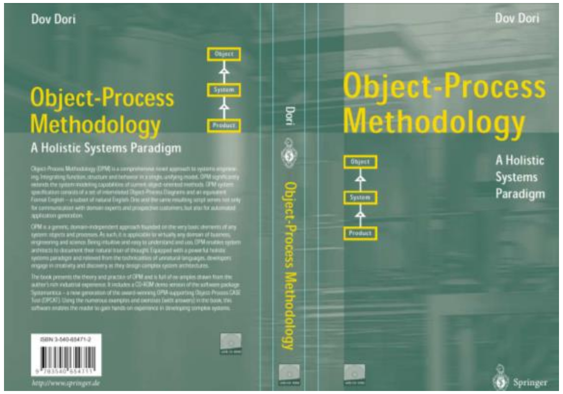
Overview

- Why Systems Modeling Languages?
  - Ontology, Semantics and Syntax
- OPM – Object Process Methodology
- SySML – Systems Modeling Language
- Modelica
- What does it mean for Systems Engineering of today and tomorrow (MBSE)?

Introduction to OPM

In order to rigorously architect and design products need a language to describe functions, form, concepts in a consistent way

- **UML 2.0**
  - http://www.omg.org/spec/UML/2.0/
  - Mainly used for software architecting
- **SysML 1.3**
  - http://www.omgsysml.org/
  - Generalization to cyber-physical systems
- **OPM**
  - Object-Process-Methodology
  - 2002, Prof. Dov Dori, Technion
  - ISO Standard 19450 (2015, new !)


Dov Dori
Object-Process Methodology
A Holistic Systems Paradigm
Object
System
Product
Dori
Object-Process Methodology
Dov Dori
Object-Process Methodology
Object
System
Product
A Holistic Systems Paradigm
Springer

Motivation for OPM

- Typical Product Representations
  - Sketches
  - Engineering Drawings
  - UML Diagrams (Software)

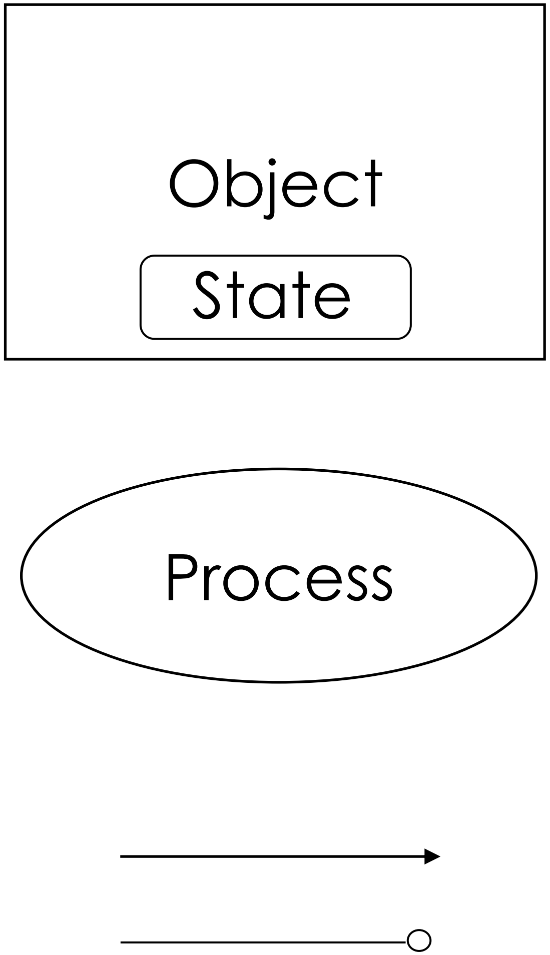
Example: Refrigerator Kenmore 2.5 cu ft

- Need for a Unified Representation
  - Show functions
  - Show function attributes
  - Show objects (operands, system components, consumables …)
  - Show object attributes
  - Show links

Object Process Methodology is a generic system modeling language that has been successfully applied to Systems Architecting of Complex Products

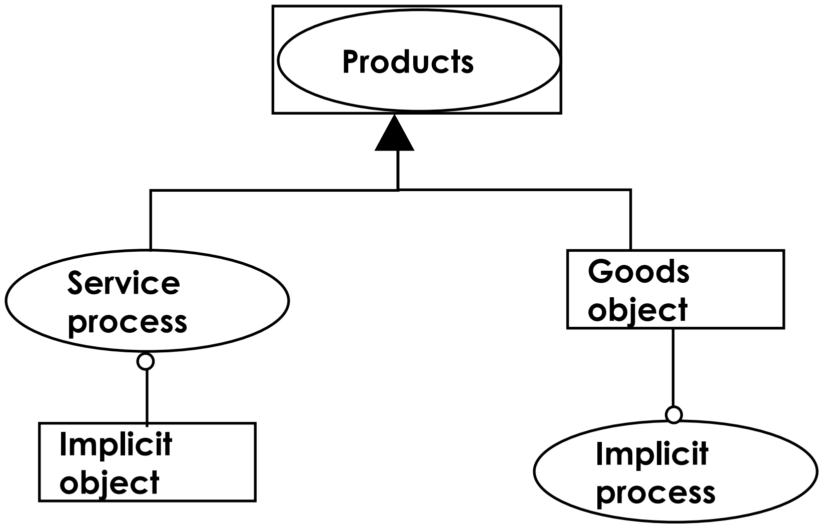
Ontology of Object Process Modeling

- **Object**: that which has the potential of stable, unconditional existence for some positive duration of time. Objects have states.
- Form is the sum of objects
- **Process**: the pattern of transformation applied to one or more objects. Processes change states.
- Function emerges from processes
- All links between objects and processes have precise semantics



OPM: Goods and Services

- Goods are objects
- Services are processes
- With every product good object, there is an implicit process which is linked to value
- With every product service process, there is always an implicit object



Product/systems always come in object-process pairs, and value is always linked to process

Structural Links in OPM

- Structural Links
  - Link Objects to Objects

→ Is related to …

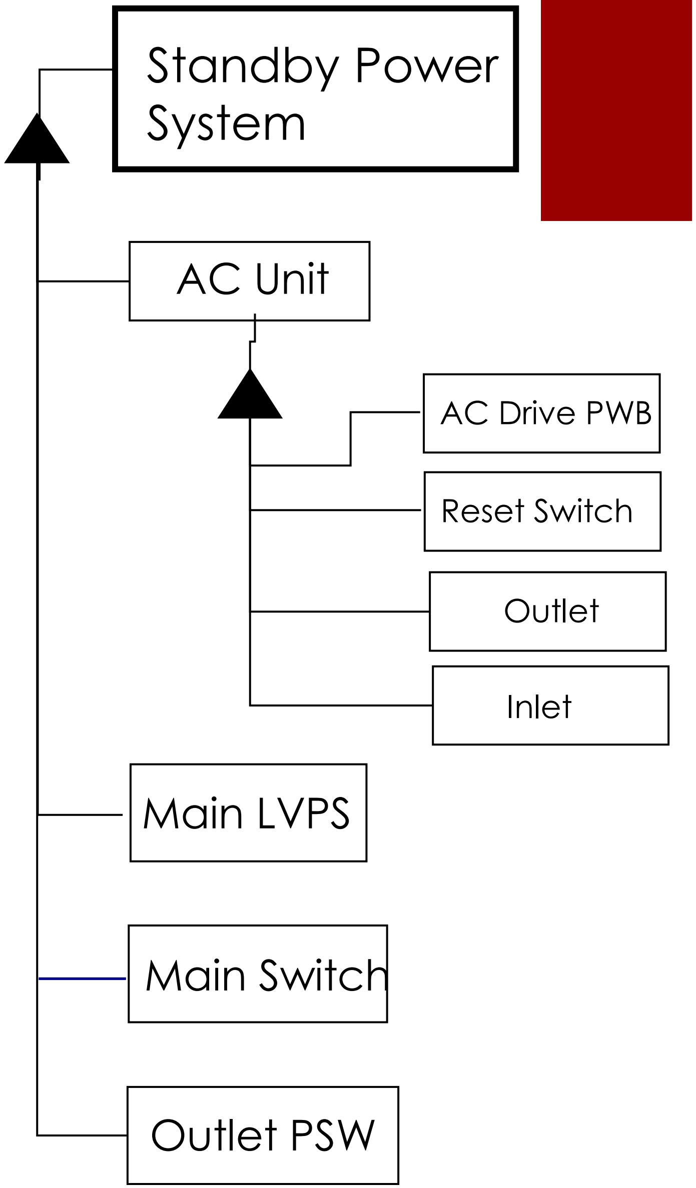
"powers" → Tagged link (suppressed process)

▲ Decomposes to, aggregates to

◬ Is characterized by, exhibits

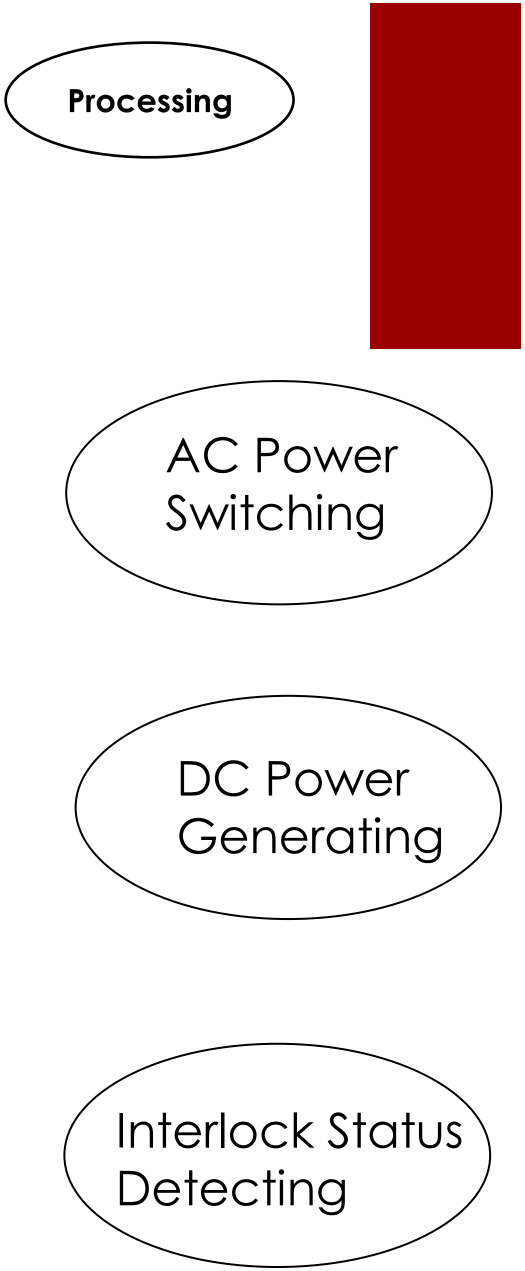
△ Specializes to, generalizes to

⟁ Instantiated to, belongs to the class of



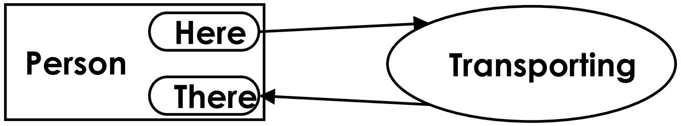
Processes

- Defined: A process is the pattern of transformation applied to one or more objects
- Cannot hold or touch a process - it is fleeting
- Generally creation, change, or destruction
  - resultee object
  - operand (its states are affected by the process)
  - consumee
- A process relies on at least one object in the pre-process set
- A process transforms at least one object in the pre-process set
- A process takes place along a time line
- A process is associated with a verb
  - Express processes in Gerund form: ____**ing**

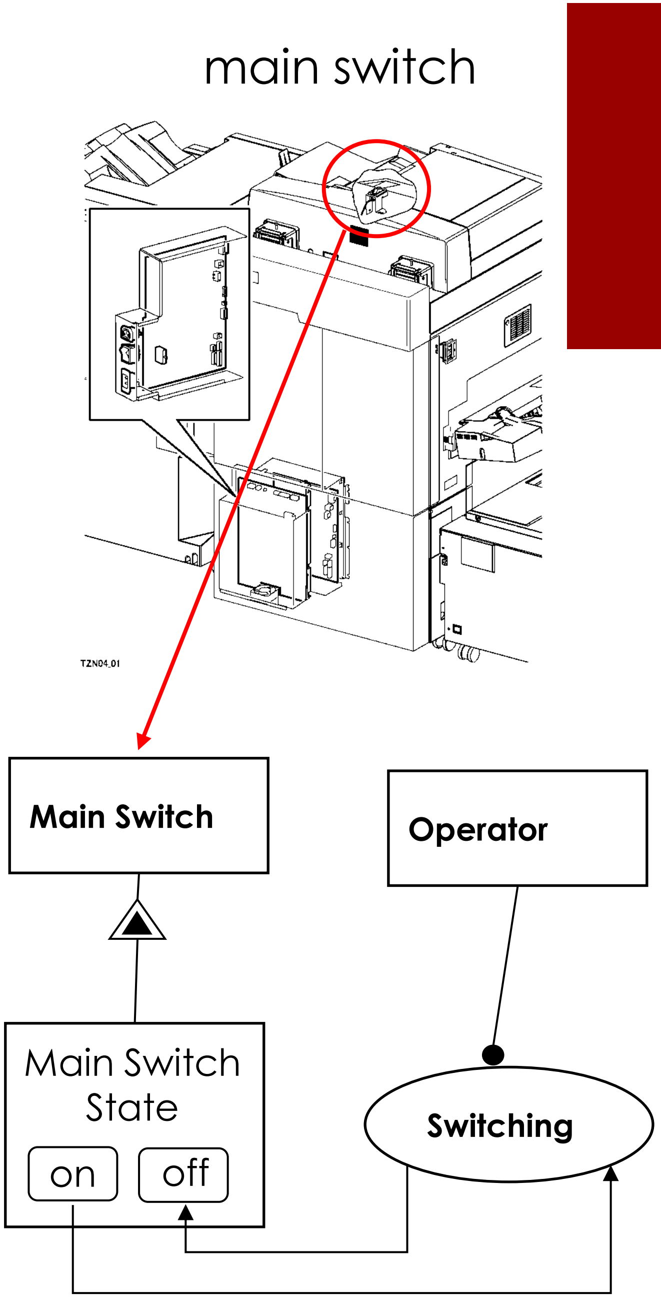


Process and its Links

- A process is associated with a verb and stateless
- There is a family of 7 types of links from process to object
- A process can change the states of its operand(s) through input and output links

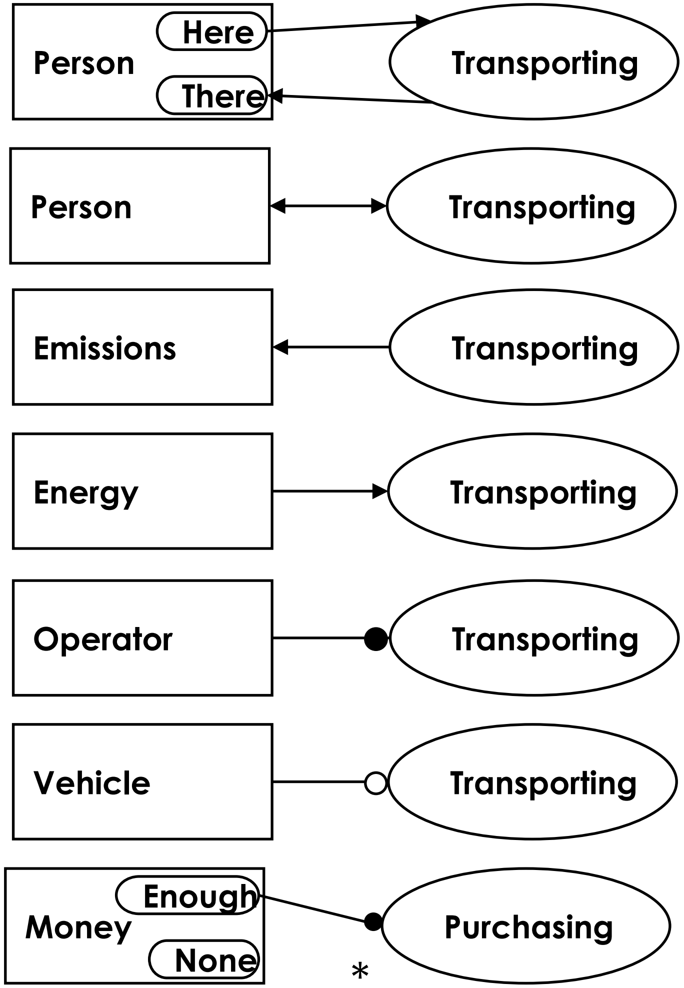


Transporting changes a person from here to there



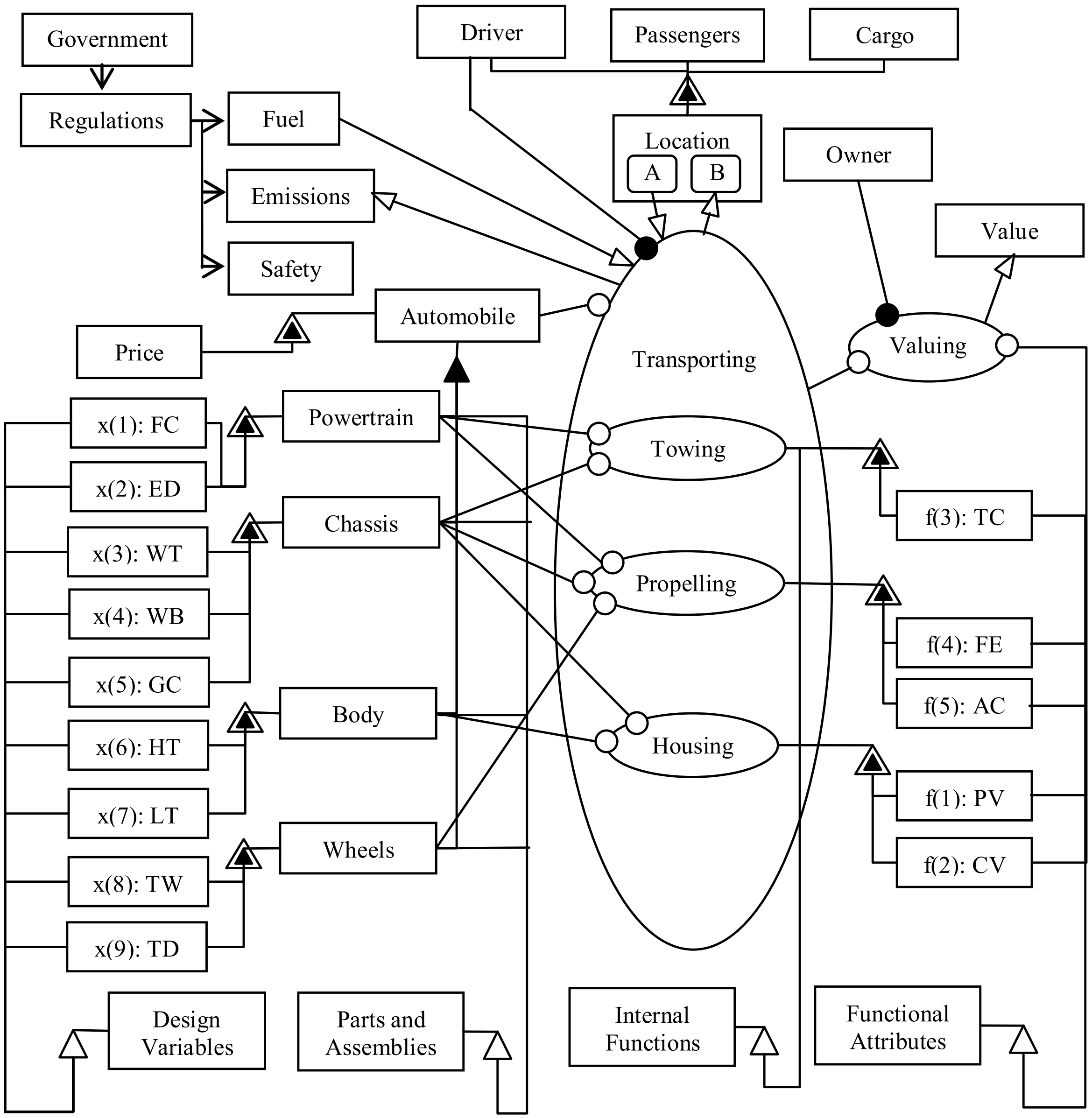
Summary Object-Process Links

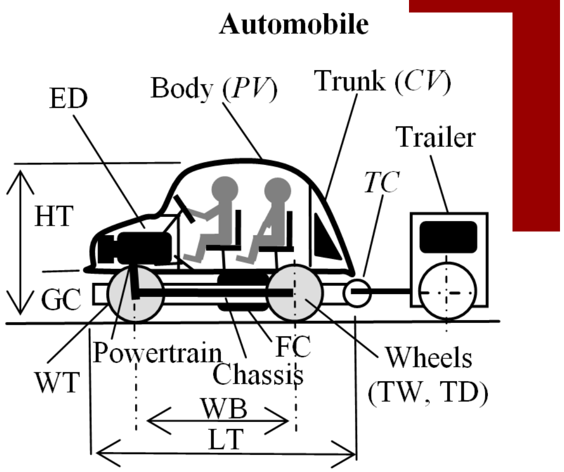
- P changes O (from state A to B).
- P affects O (affectee)
- P yields O (resultee)
- P consumes O (consumee)
- P is handled by O (agent)
- P requires O (instrument)
- P occurs if O is in state A



* conditional link also shown as ©————

High Level OPM of a Car





This view shows all main elements of the car as a product system:

-objects

- operands
- instruments
- consumees, resultees
- operator

-processes

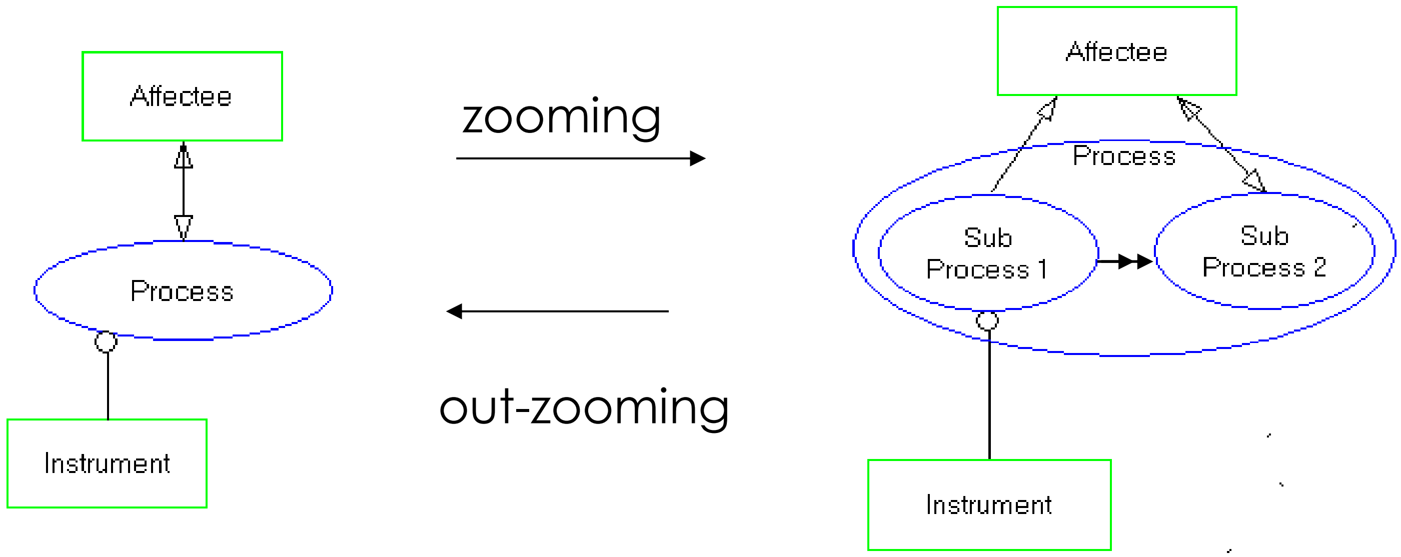
-attributes

-x: design variables

-f: functional behavior

Managing Complexity in OPM

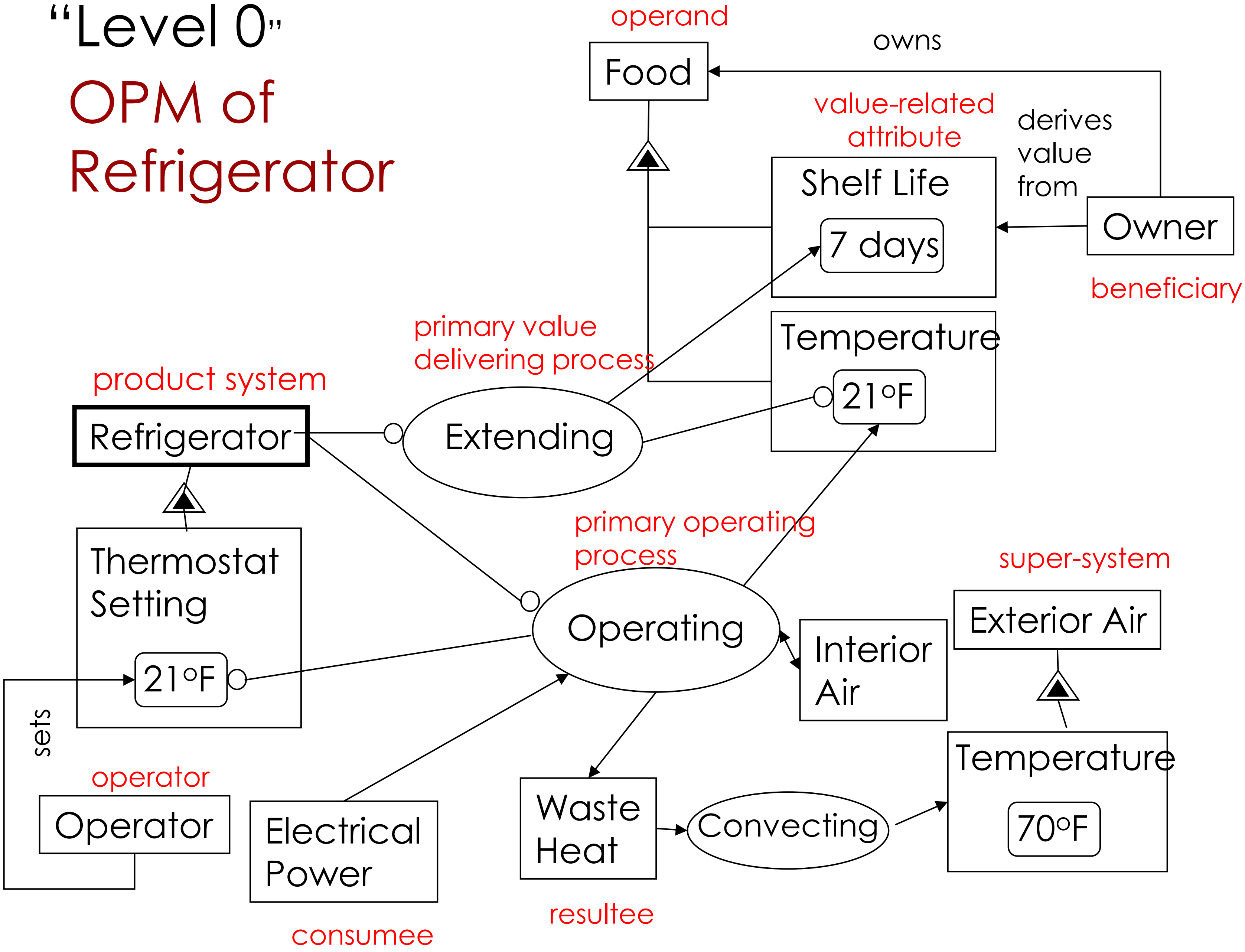
- OPM has three mechanism for managing system complexity:
  - unfolding/folding is used for refining/abstracting the structural hierarchy of an object;
  - in-zooming/out-zooming exposes/hides the inner details of a process within its frame;
  - state expressing/suppressing exposes/hides the states of an object.

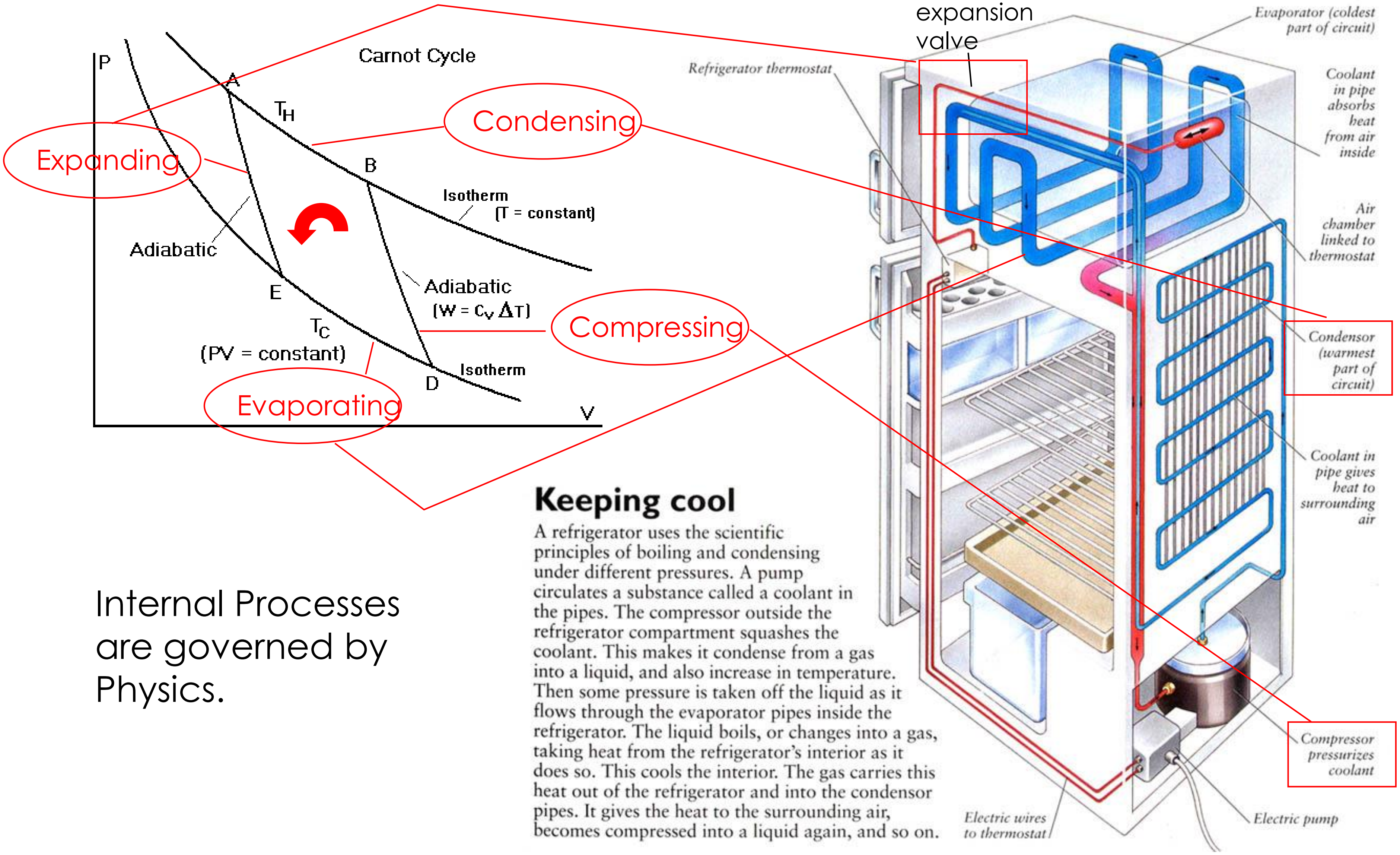


zooming

out-zooming

"Level 0" OPM of Refrigerator




Carnot Cycle
P
V
A
B
D
E
T_H
T_C
Isotherm
(T = constant)
Isotherm
Adiabatic
Adiabatic
(W = C_V ΔT)
(PV = constant)
Expanding
Condensing
Compressing
Evaporating
expansion valve
Refrigerator thermostat
Evaporator (coldest part of circuit)
Coolant in pipe absorbs heat from air inside
Air chamber linked to thermostat
Condensor (warmest part of circuit)
Coolant in pipe gives heat to surrounding air
Compressor pressurizes coolant
Electric wires to thermostat
Electric pump

Keeping cool

A refrigerator uses the scientific principles of boiling and condensing under different pressures. A pump circulates a substance called a coolant in the pipes. The compressor outside the refrigerator compartment squashes the coolant. This makes it condense from a gas into a liquid, and also increase in temperature. Then some pressure is taken off the liquid as it flows through the evaporator pipes inside the refrigerator. The liquid boils, or changes into a gas, taking heat from the refrigerator's interior as it does so. This cools the interior. The gas carries this heat out of the refrigerator and into the condensor pipes. It gives the heat to the surrounding air, becomes compressed into a liquid again, and so on.

Internal Processes are governed by Physics.

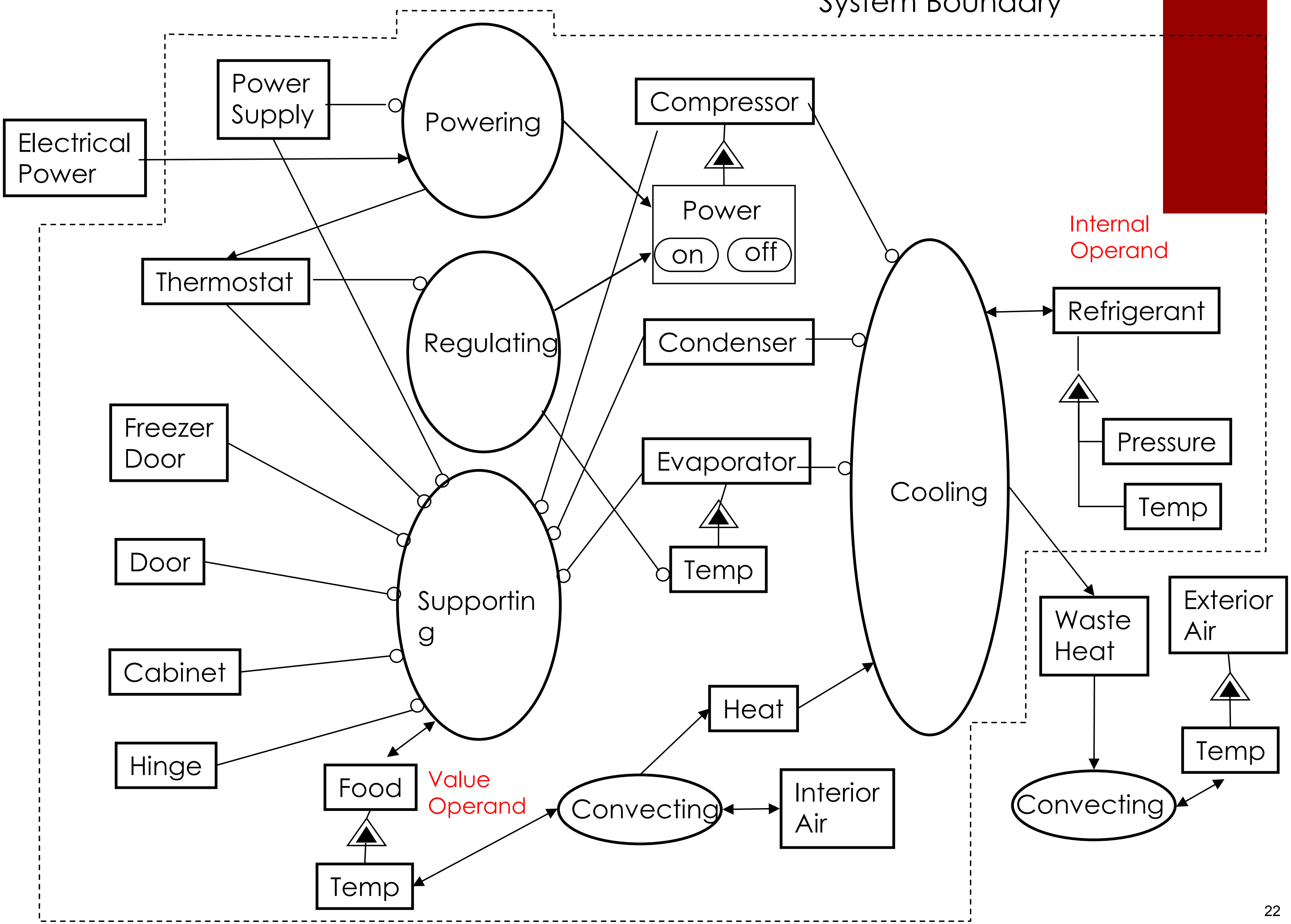
Refrigerator: Functional Decomposition

Level -1 (4)
Level -2 (15)

Operating

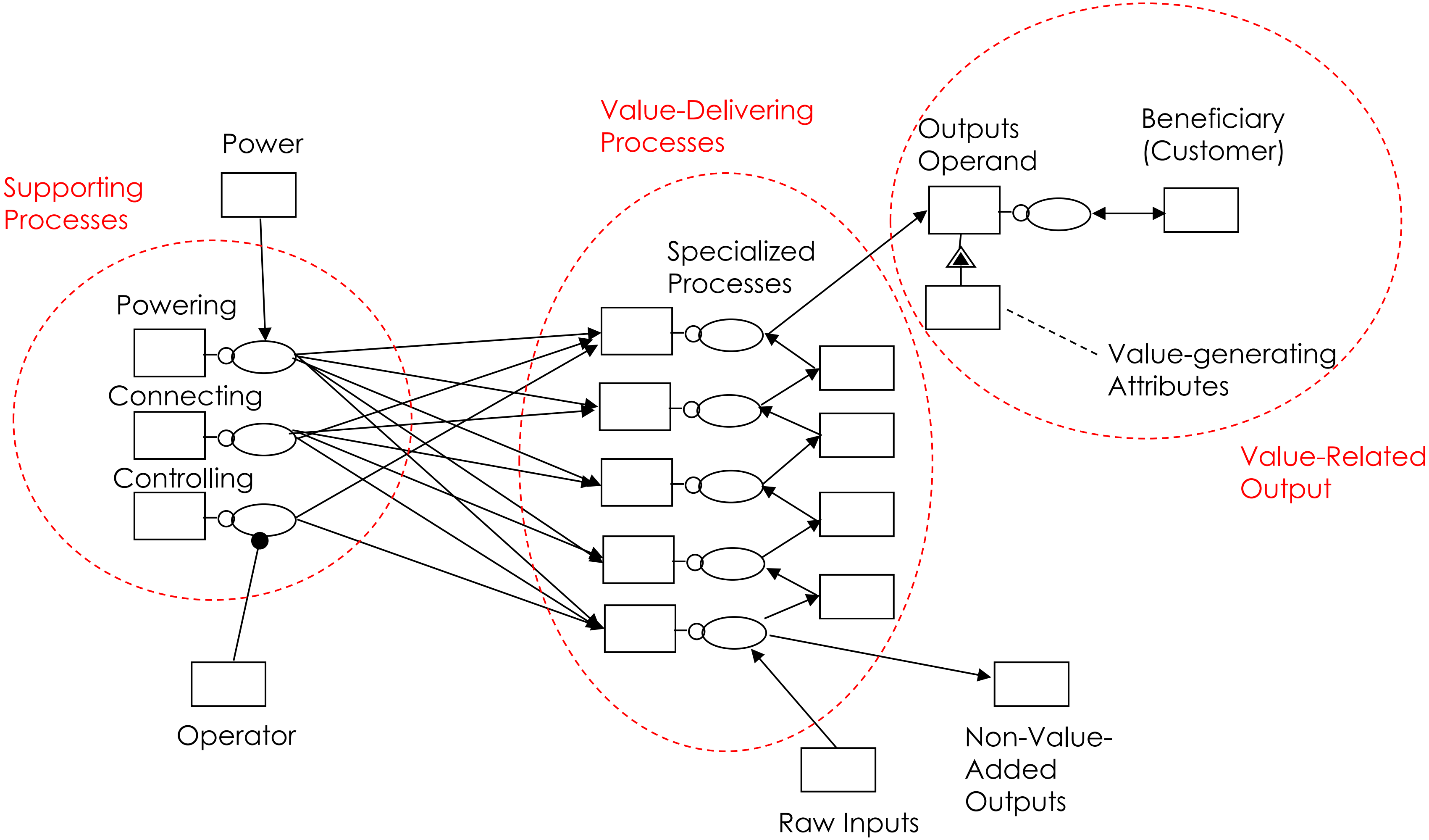
- Powering
  - Grounding
  - Protecting
  - Supplying
- Regulating
  - Sensing
  - Switching
  - Setting
- Cooling
  - Expanding
  - Evaporating
  - Compressing
  - Condensing
  - Absorbing
- Supporting
  - Opening
  - Closing
  - Retaining
  - Connecting

OPM of Refrigerator Level -1



Generic System OPM

OPMs of most complex opto-mechanical systems look like this



How to generate a System OPM

Top-Down

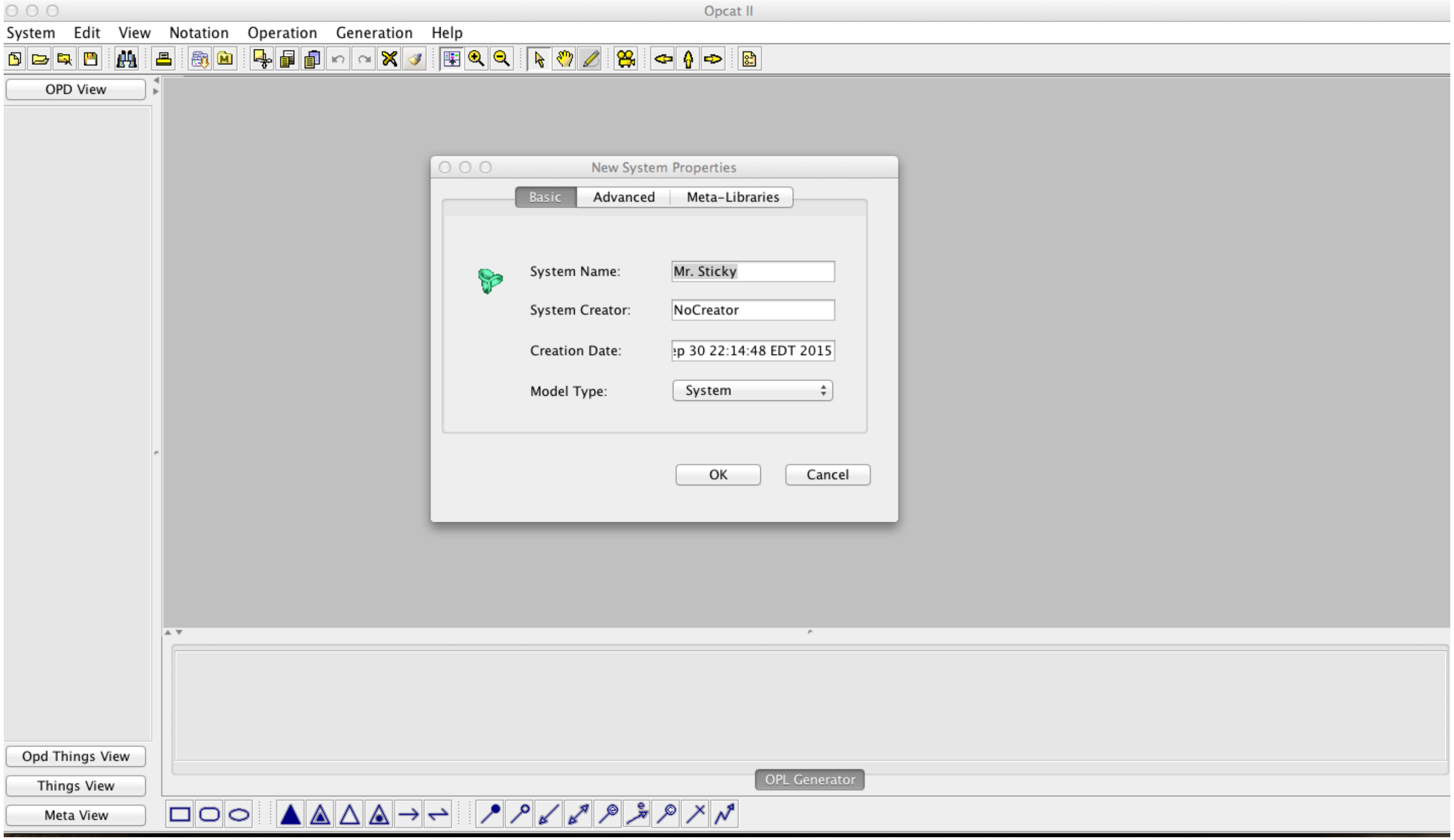
- Start with the stakeholder(s) (customer in mind)
- Map value delivery process(es) at Level 0
- Get to greater levels of detail in layers
- This is **system/product architecting** !
  - Reduce Ambiguity
  - Focus Creativity
  - Manage Complexity

Bottom-Up

- Decompose form of existing product or design (product dissection):
  - Parts List/BOM
- Generate an initial product decomposition
- Assign processes to elements of form
- Complete initial OPM and iterate
- This is **reverse engineering** !

OPCAT Demo

OPCAT is a Java-based software to generate OPM Models



System Name: Mr. Sticky

System Creator: NoCreator

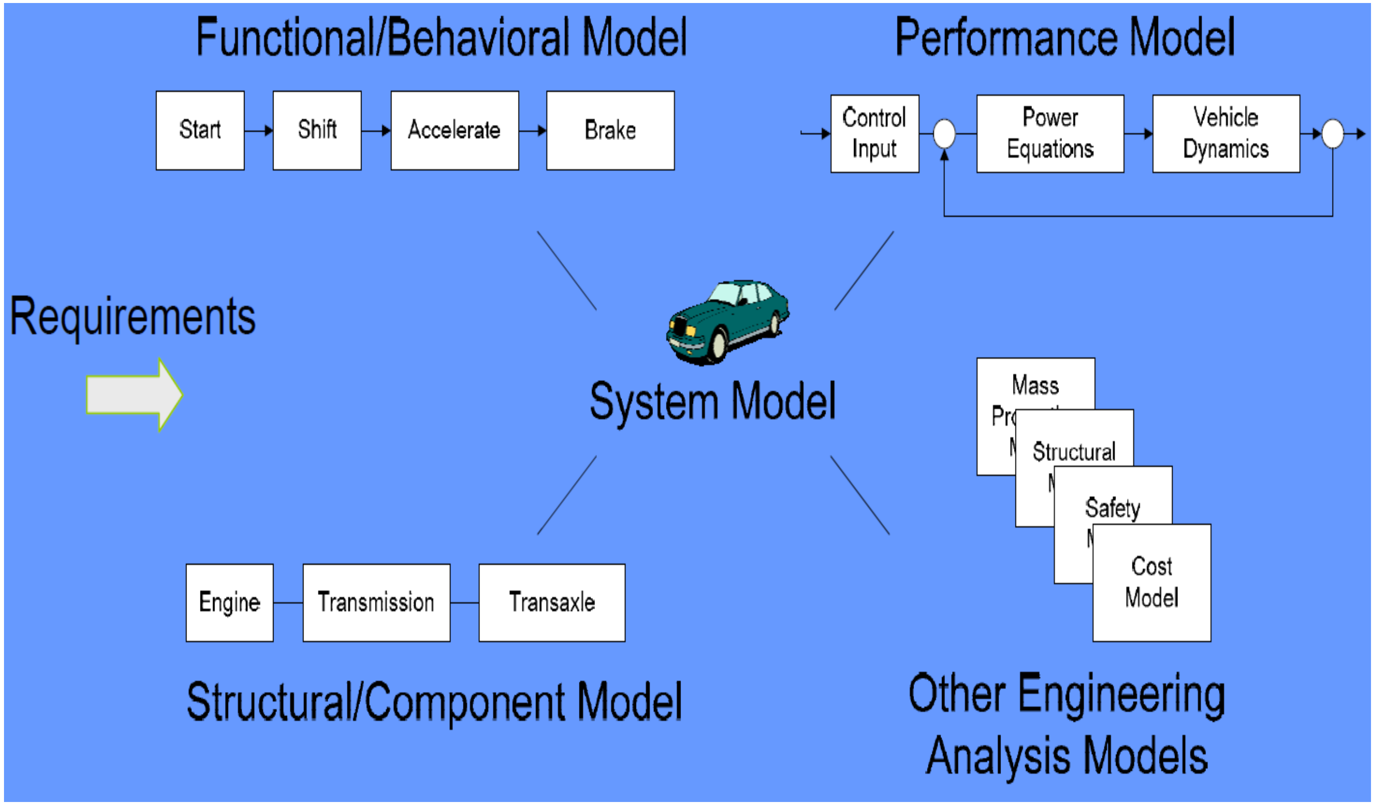
Creation Date: :p 30 22:14:48 EDT 2015

Model Type: System

Overview

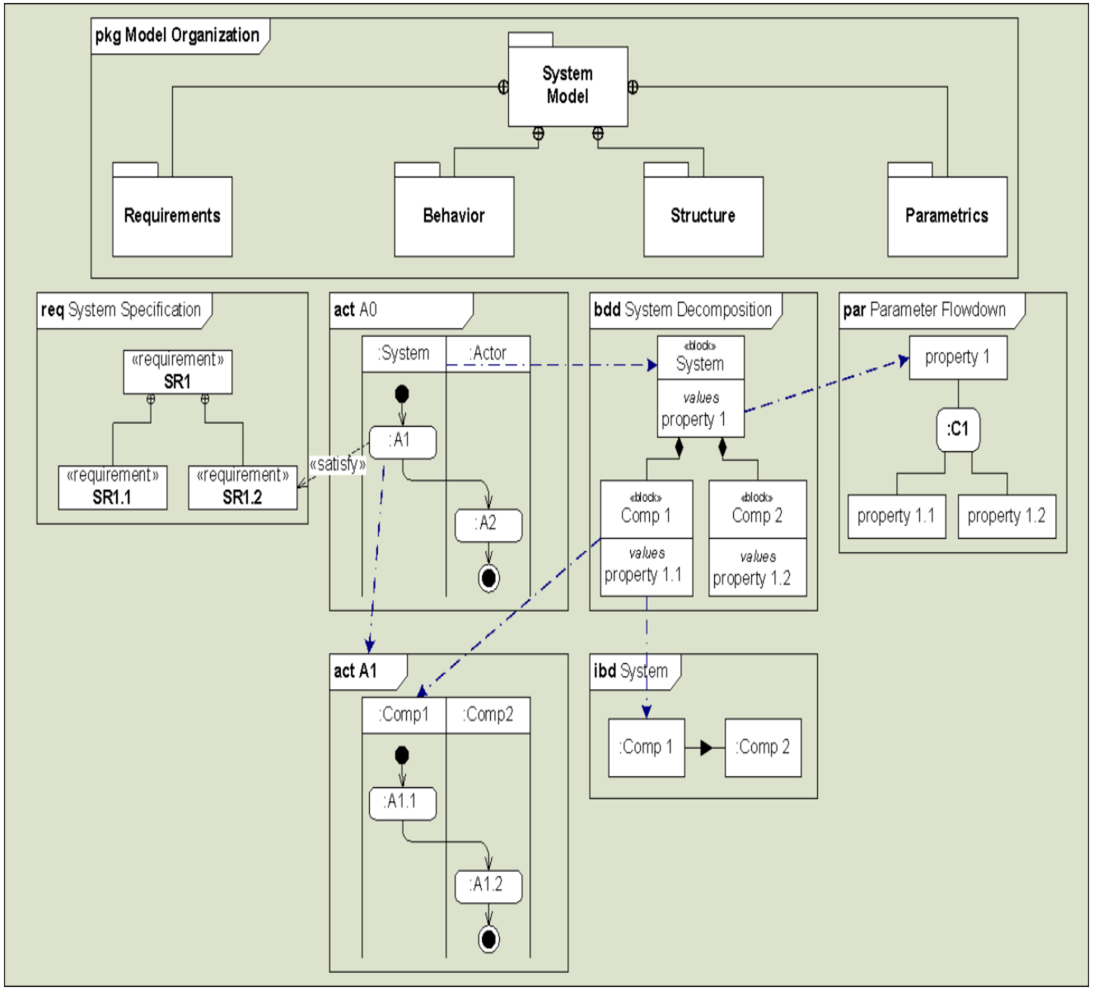
- Why Systems Modeling Languages?
  - Ontology, Semantics and Syntax
- OPM – Object Process Methodology
- SySML – Systems Modeling Language
- Modelica
- What does it mean for Systems Engineering of today and tomorrow (MBSE)?

MBSE System Modeling Scope



System model must capture information about all aspects of system.

The Systems Modeling Language



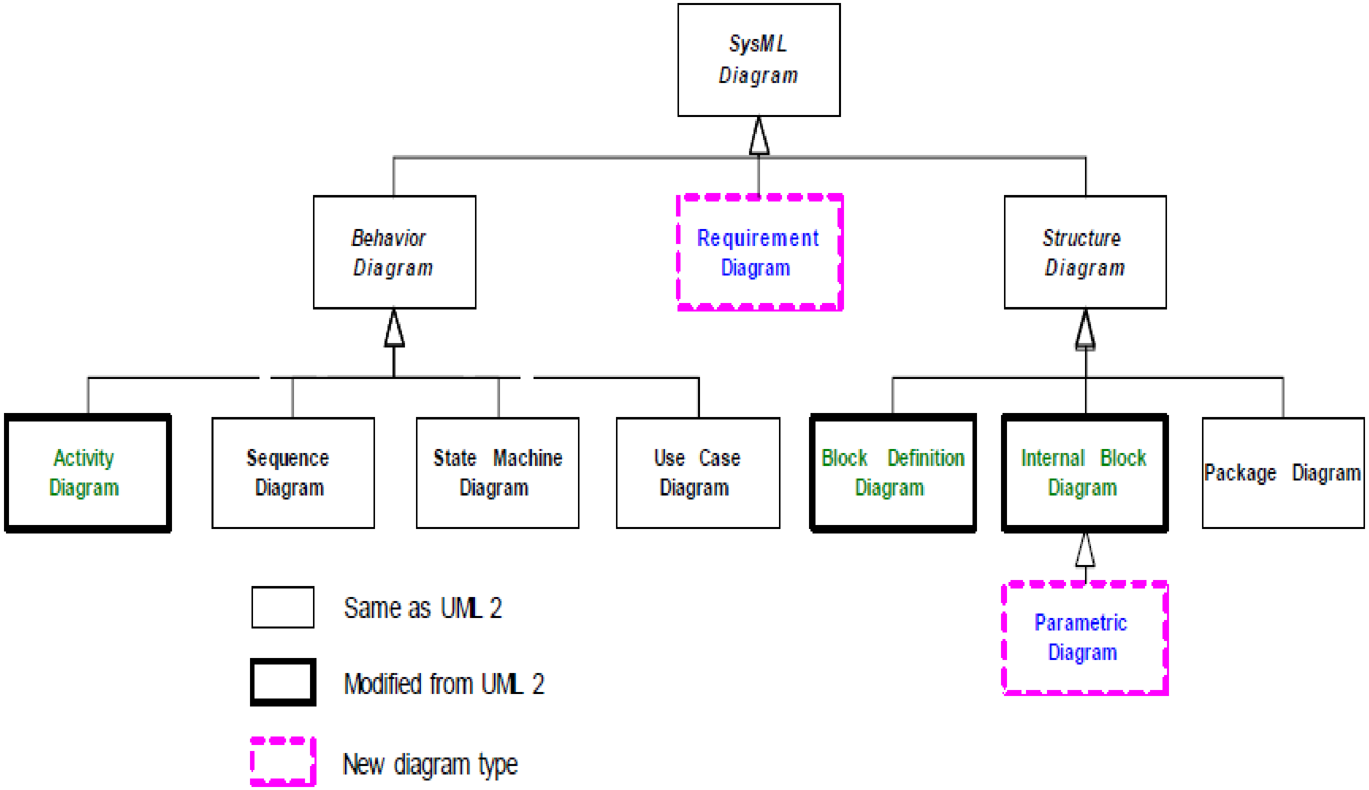
SysML diagrams capture different types of system information. Diagrams can be linked together

SysML created starting in 2001 by OMG/INCOSE.

Applications of SysML

- Requirements engineering
  - Implement requirements as constraints on the model, instead of as text statements within the model
- System Description
  - Using SysML allows study of potentially more mission concepts within the same timeframe
- Integration with Analysis Tools
  - Graph transformations to support dynamic analysis in Simscape™
  - Integration with Phoenix ModelCenter® allows analysis in a range of tools

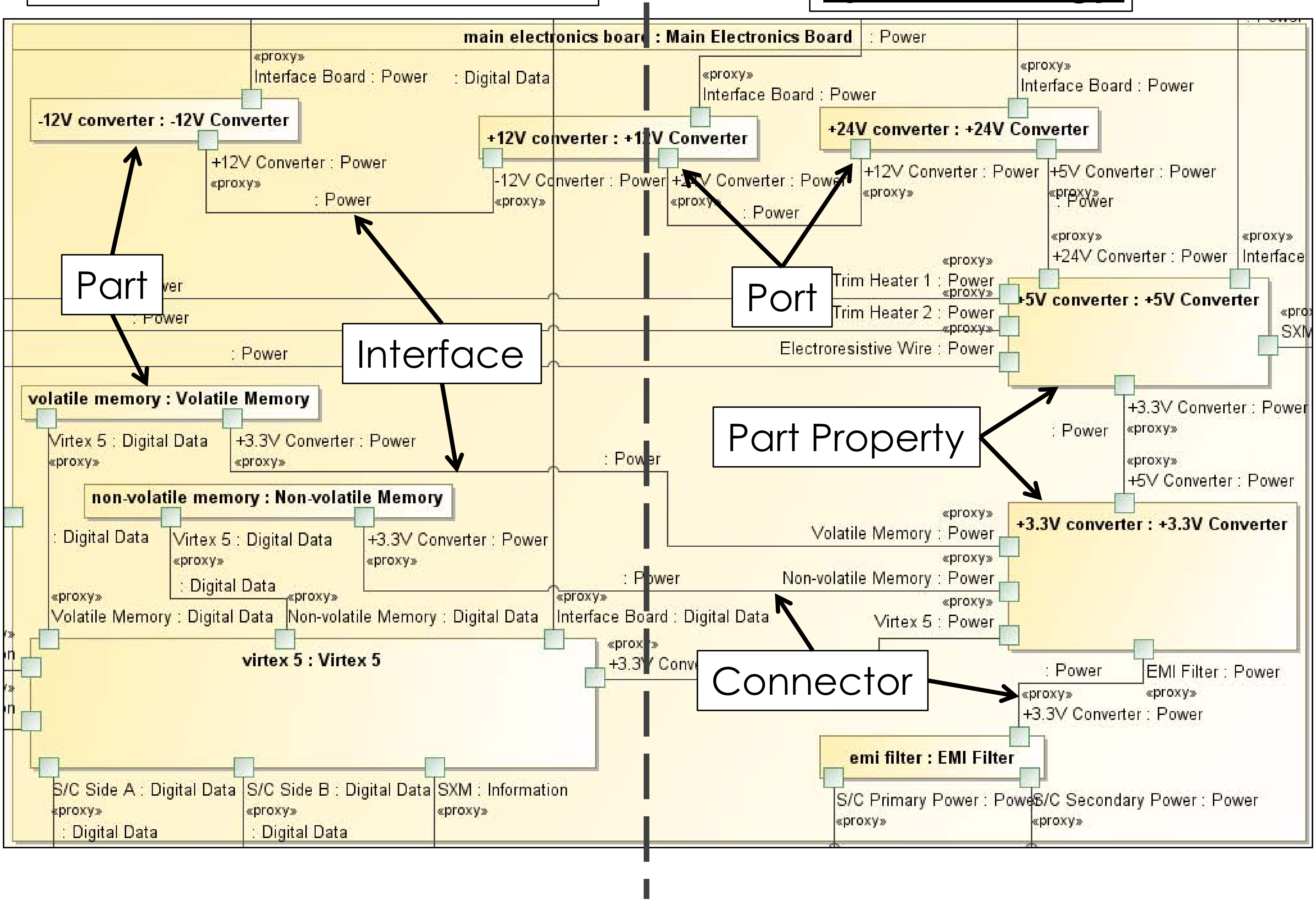
SysML Diagram Hierarchy



The types of SysML diagrams

System Engineering Ontology

SysML Ontology



Part

Interface

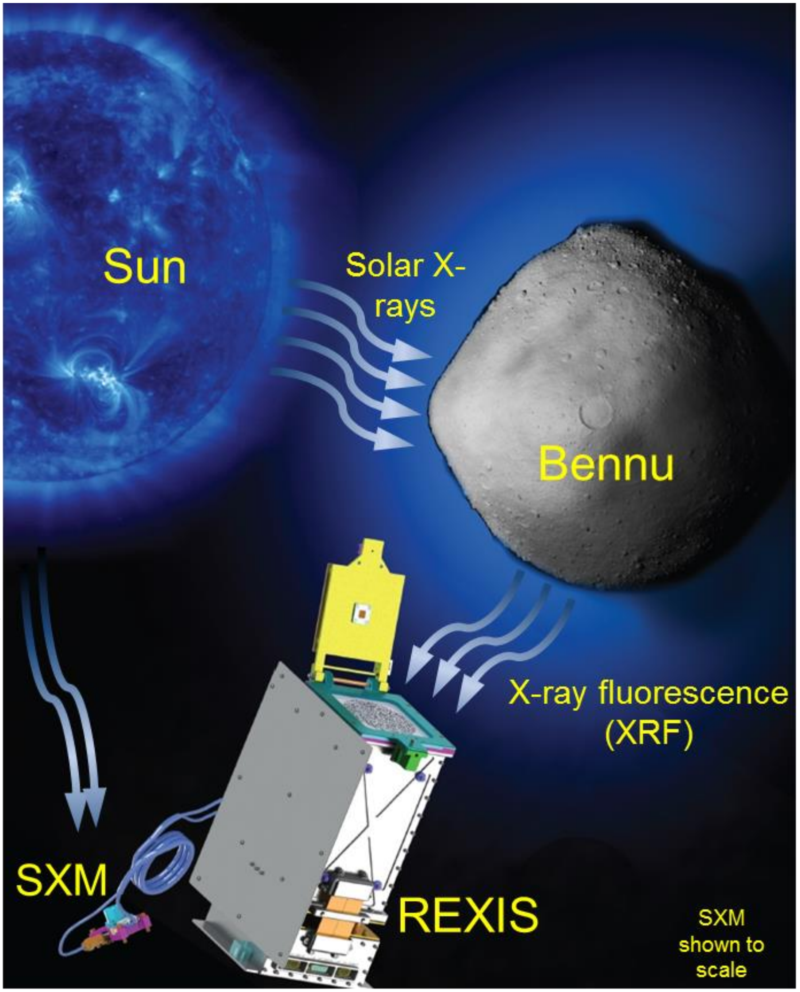
Port

Part Property

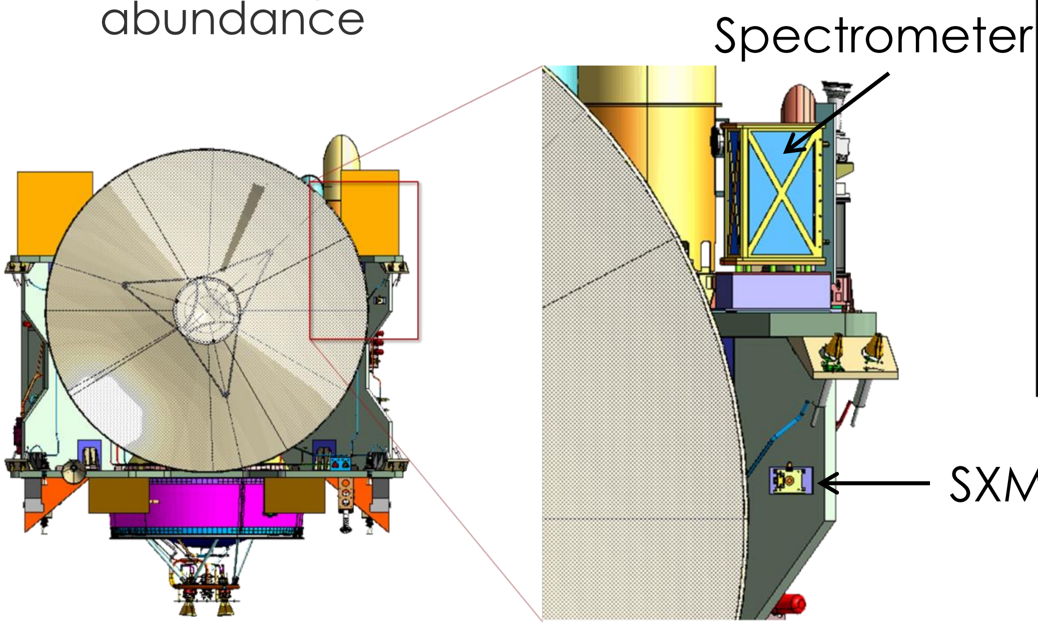
Connector

Case Study: REXIS

- One of five instrument on the OSIRIS-REx asteroid sample return mission scheduled for launch in 2016
- Measures X-rays that are fluoresced from Bennu
- Fluorescent line energies depend on the electronic structure of the matter
  - Provides a unique elemental signature
  - Line strengths reflect element abundance

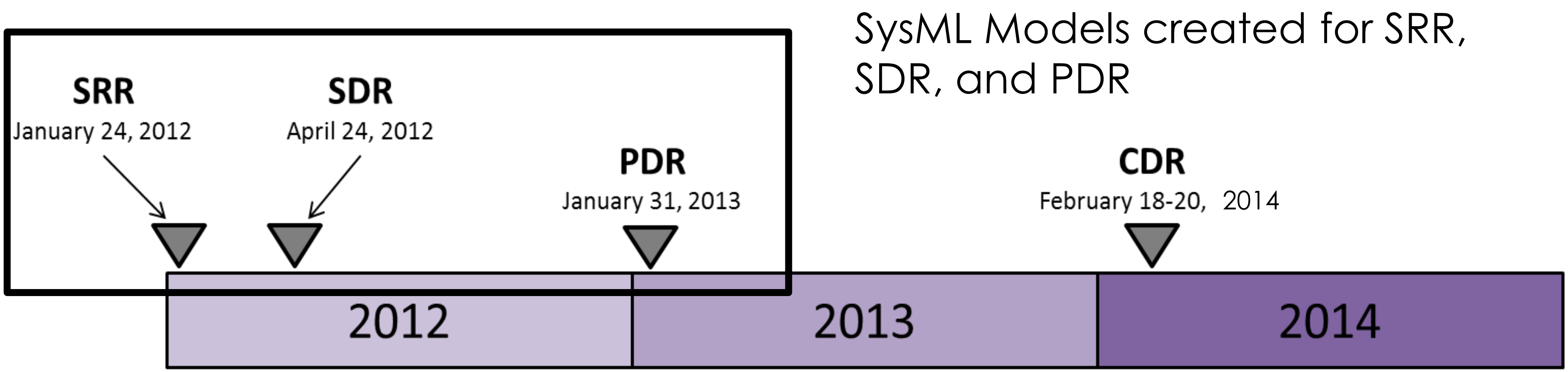


These images are in the public domain.



These images are in the public domain.

REXIS Design History Overview

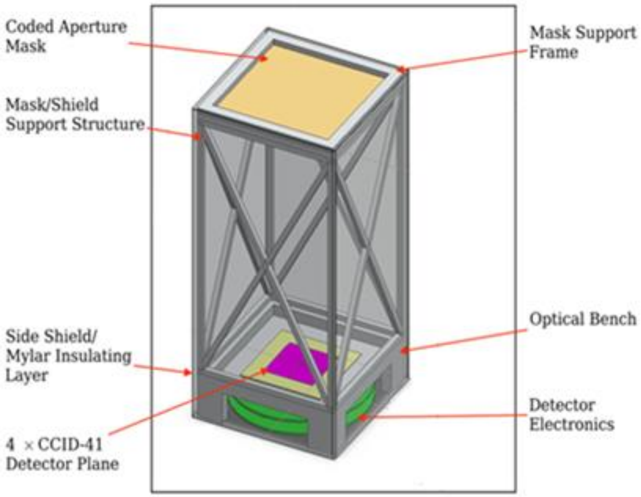


SysML Models created for SRR, SDR, and PDR

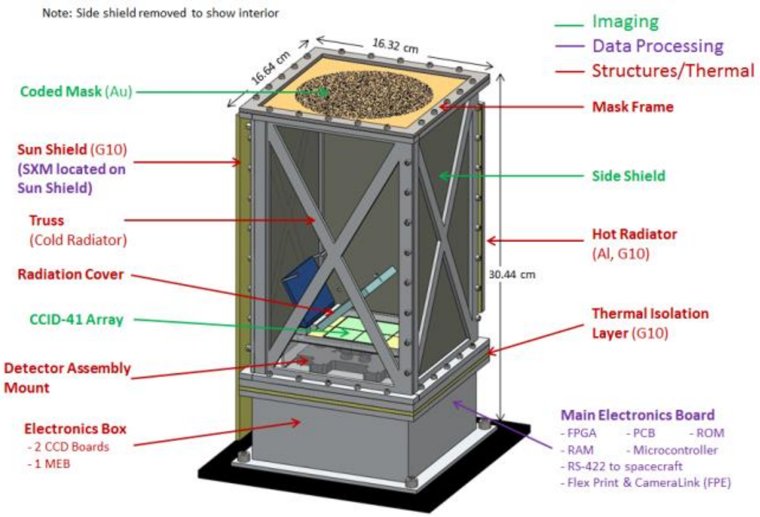
- SysML models created at SRR, SDR, and PDR

- From Fall 2011 through Spring 2012, REXIS team composed primarily of undergraduates
  - With grad students and faculty mentors
- From Summer 2012 to present, REXIS team composed primarily of grad students
  - With faculty mentors and undergraduate volunteers

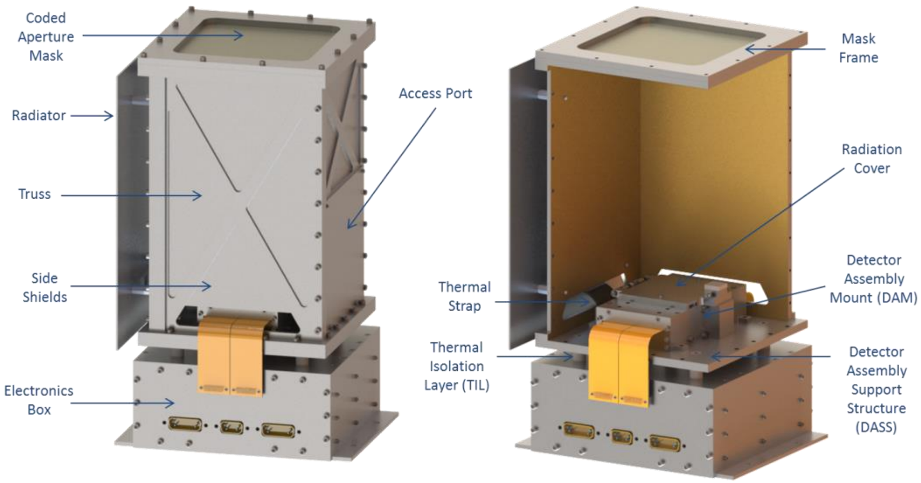
REXIS Design History


Coded Aperture Mask
Mask Support Frame
Mask/Shield Support Structure
Optical Bench
Side Shield/ Mylar Insulating Layer
Detector Electronics
4 × CCID-41 Detector Plane

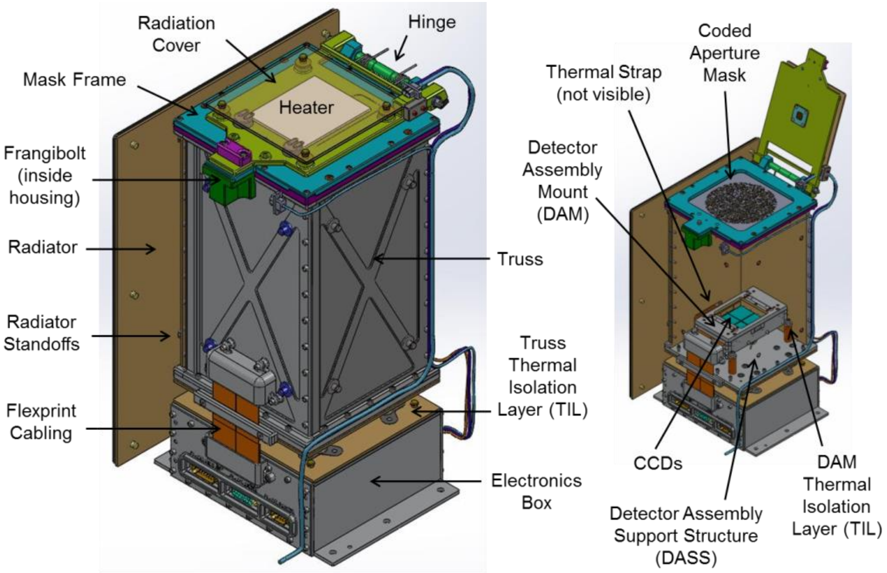
SRR - January 2012



SDR - April 2012



PDR - January 2013

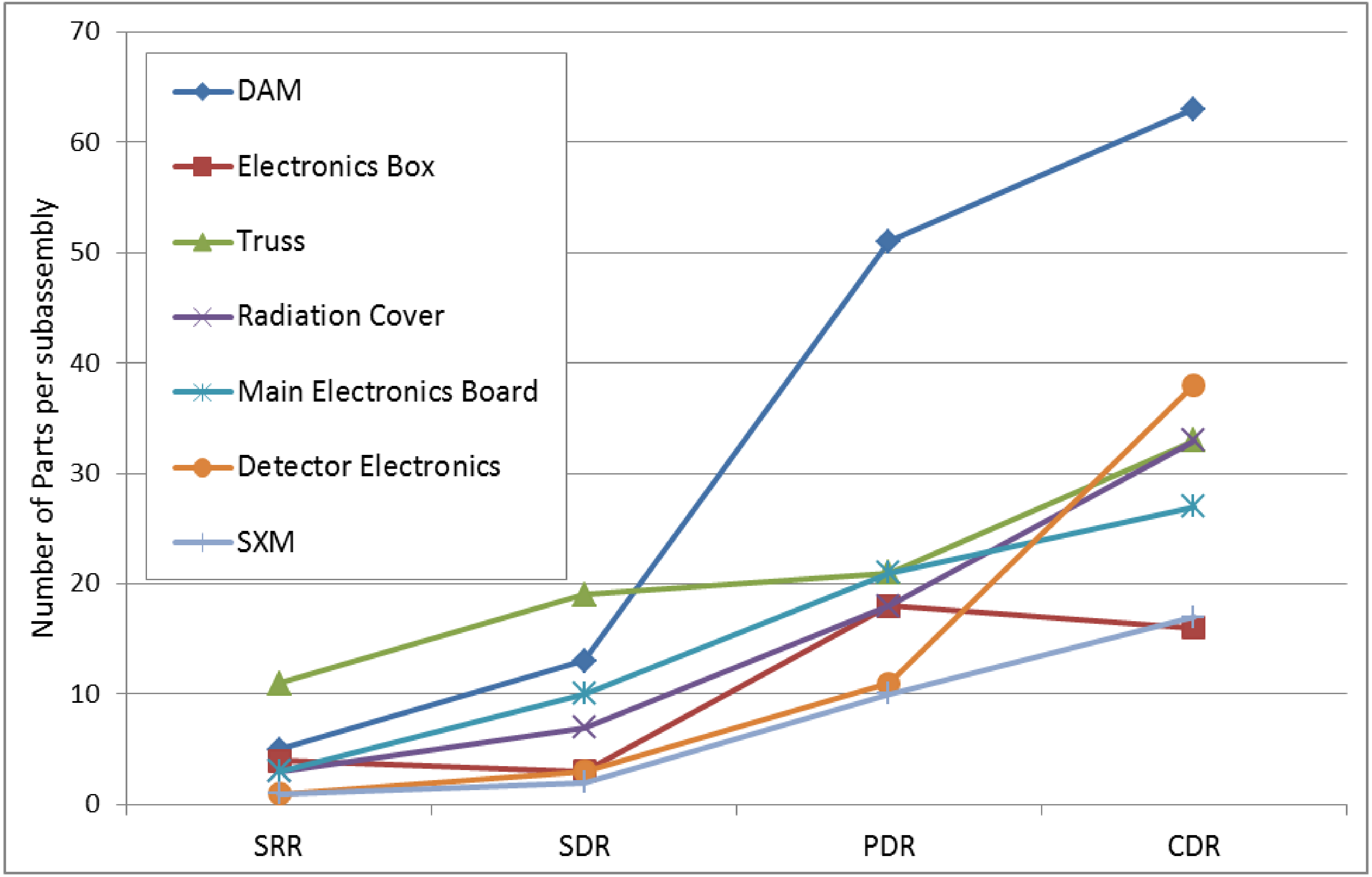

Radiation Cover
Hinge
Mask Frame
Heater
Frangibolt (inside housing)
Radiator
Radiator Standoffs
Flexprint Cabling
Truss
Truss Thermal Isolation Layer (TIL)
Electronics Box
Thermal Strap (not visible)
Coded Aperture Mask
Detector Assembly Mount (DAM)
CCDs
Detector Assembly Support Structure (DASS)
DAM Thermal Isolation Layer (TIL)

CDR - February 2014

These images are in the public domain.

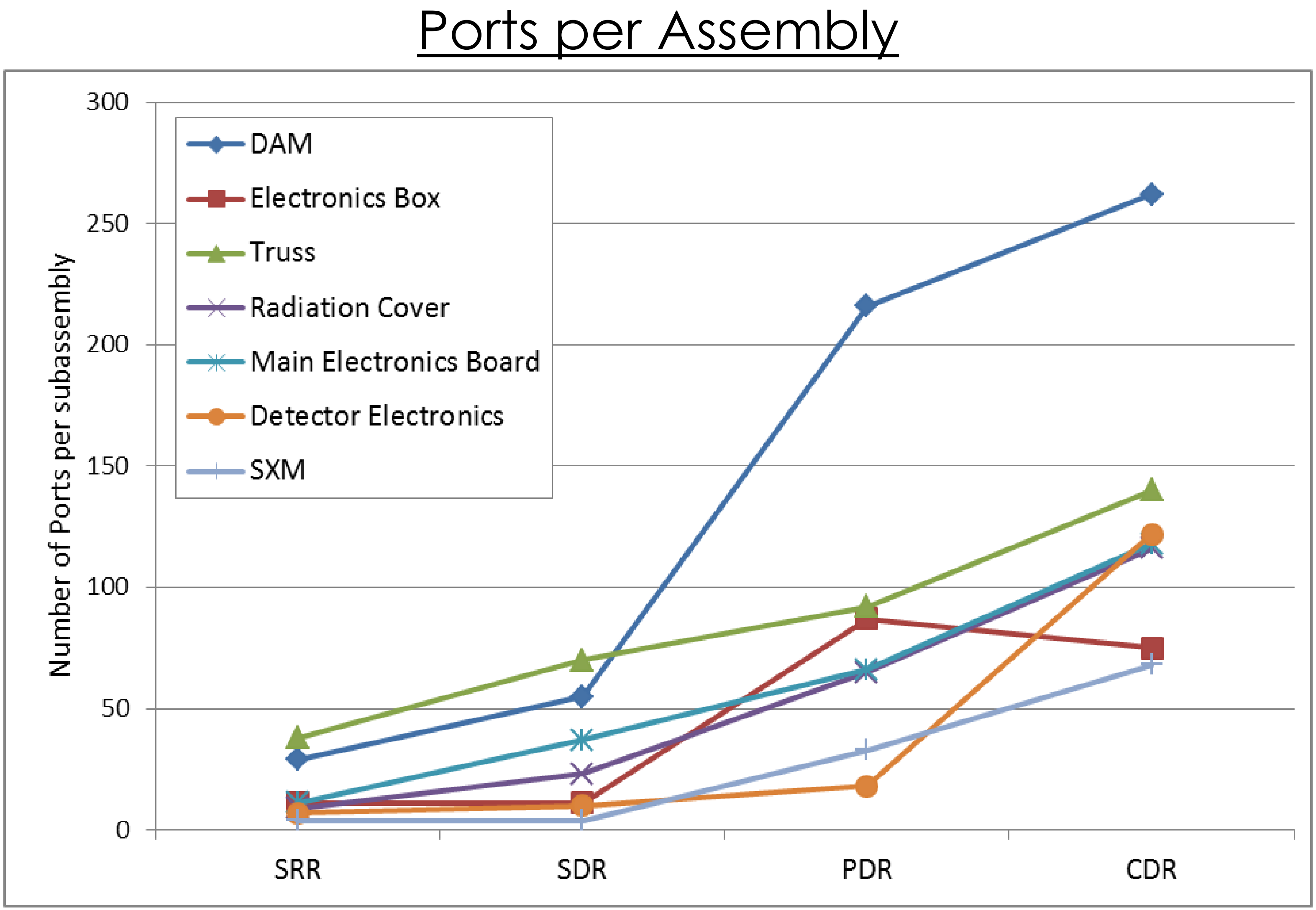
Design History Statistics

Parts per Assembly



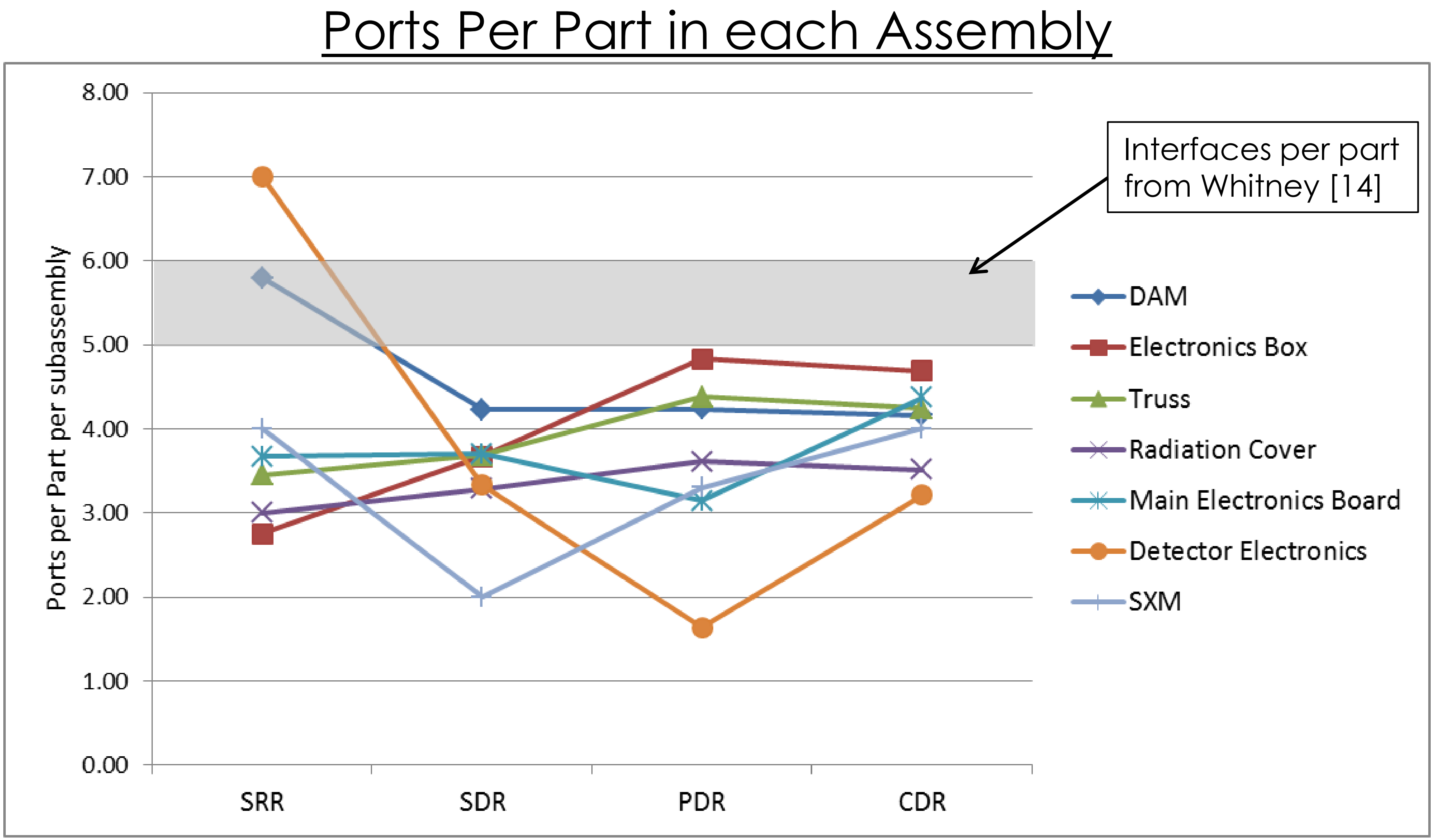
All assemblies experienced parts growth

Design History Statistics



All assemblies experienced interface growth

Design History Statistics



Slightly fewer interfaces per part than other systems in the literature

SySML – System Modeling Language

- SysML Demo (Mark Chodas)

Overview

- Why Systems Modeling Languages?
  - Ontology, Semantics and Syntax
- OPM – Object Process Methodology
- SySML – Systems Modeling Language
- Modelica
- What does it mean for Systems Engineering of today and tomorrow (MBSE)?

Introduction to Cyber-Physical System Modeling in Modelica

Modelica Language

Modelica is a language designed to enable mathematical modeling of cyber-physical systems

Declarative
Equations and mathematical functions allow **acausal** modeling, high level specification and increased correctness (define the problem rather than how it needs to be solved)

Multi-domain modeling
Combines components from electrical, mechanical, thermodynamic, hydraulic, biological, control, event, real-time and custom domains etc...

Everything is a class
Strongly typed object-oriented language with a general class concept, Java & MATLAB-like syntax

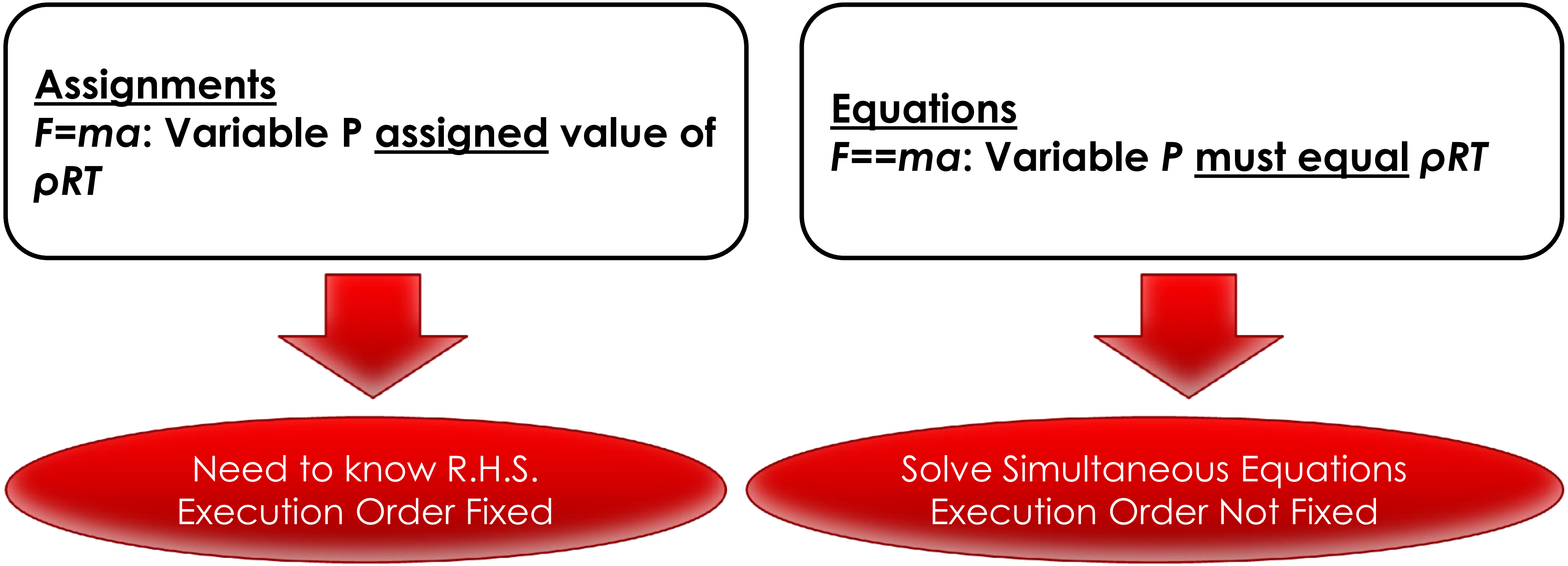
Visual component programming
Hierarchical system architecture capabilities

Efficient, non-proprietary
Efficiency comparable to C; advanced equation compilation,
e.g. 300 000 equations, ~150 000 lines on standard PC

Taken with permission from Professor Peter Fritzson

Acausal Modeling

Linking components via energy, mass, information flows etc. without specifying the directionality of connections.



Assignments
$F=ma$: **Variable P assigned value of** ρRT

Equations
$F==ma$: **Variable** P **must equal** ρRT

Need to know R.H.S.
Execution Order Fixed

Solve Simultaneous Equations
Execution Order Not Fixed

Acausal Modeling

A component model generally consists of:

1. Connection points or “Ports” in mechanical, thermal, electrical or custom domains (connections can only be made between ports of the same domain).

1. Variables and Parameters

1. Governing Equations

Modelica Language

Capacitor Example

```
connector Pin
Real v;
flow Real i;
end Pin;

model Capacitor

parameter Real C;
Pin p, n;
Real u;

equation
0 = p.i + n.i;
 u = p.v – n.v;
C*der(u) = p.i;

end Capacitor;
```

Create Port Type Pin That Carries Current and Voltage Variables.

“flow” indicates that the sum or all currents into a node is 0. Kirchhoff’s Current Law Holds. Mass flow is also a “flow” variable

Capacitance

p and n hold current and voltage information at the capacitor’s ports

Voltage across capacitor (internal variable)

Governing Equations

Modelica Language

Pressure Loss Example

Nominal mass flow

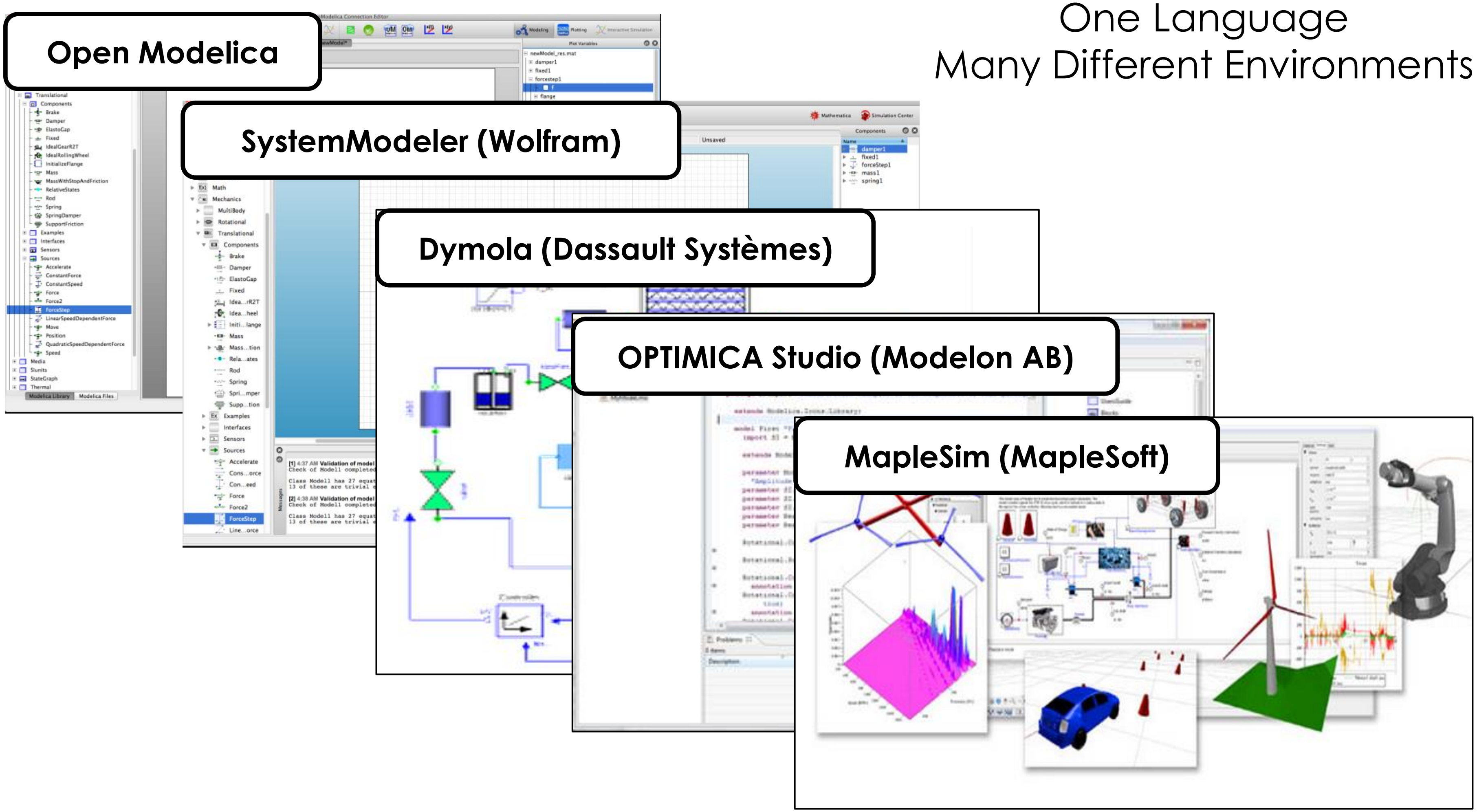
Nominal pressure drop

```
model PressureLoss
import SI = Modelica.SIunits;
...
parameter SI.MassFlowRate m_flow_nominal;
parameter SI.Pressure dp_nominal "Nominal pressure drop";
SI.Density rho "Upstream density";
SI.DynamicViscosity lambda "Upstream viscosity";
equation
...
m_flow = homotopy(actual = turbulentFlow_dp(dp, rho, lambda),
simplified = dp/dp_nominal*m_flow_nominal);

...
end PressureLoss;
```

Initialize with nominal linear calculation and then transform to full nonlinear

Modelica Environments

One Language
Many Different Environments



Open Modelica

SystemModeler (Wolfram)

Dymola (Dassault Systèmes)

OPTIMICA Studio (Modelon AB)

MapleSim (MapleSoft)

Modelica Environments

Commercial

Dymola (Dassault Systèmes)
Vertex (deltatheta)
Converge (deltatheta)
Modelica SDK (deltatheta)
MOSILAB (Fraunhofer FIRST)
SimulationX (ITI GmbH)
LMS Imagine.Lab AMESim (LMS)
MapleSim (MapleSoft)
MathCore (MathModelica)
SystemModeler (Wolfram)
OPTIMICA Studio (Modelon AB)

Free

JModelica.org
Modelicac
SimForge
OpenModelica

Matlab/Simscape Environment

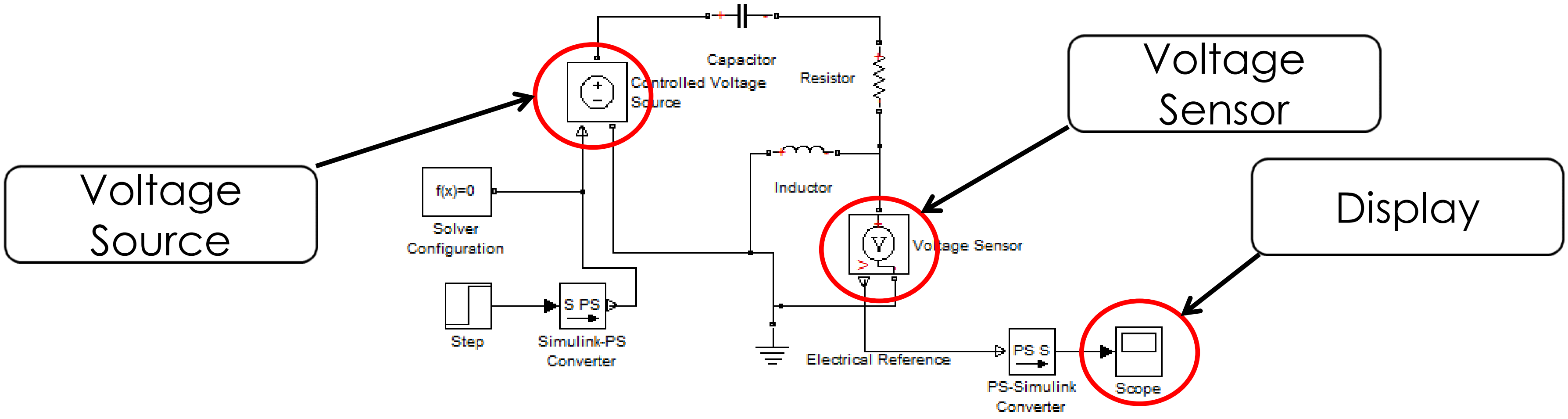
The Matlab based Simscape Physical Modeling Environment (Language Similar To Modelica)

- Built in foundation libraries of Electrical, Hydraulic, Magnetic, Mechanical, Physical Signal, Pneumatic and Thermal components.

- There are also extensions which allow some more detailed simulation of gearing, hydraulic, mechanical/robotic and power systems.

- The facilities exist to generate custom components and domains.

- Models have almost **1-1 mapping to the physical systems they describe and generally are easily reconfigurable** making the tool very intuitive (RLC circuit)

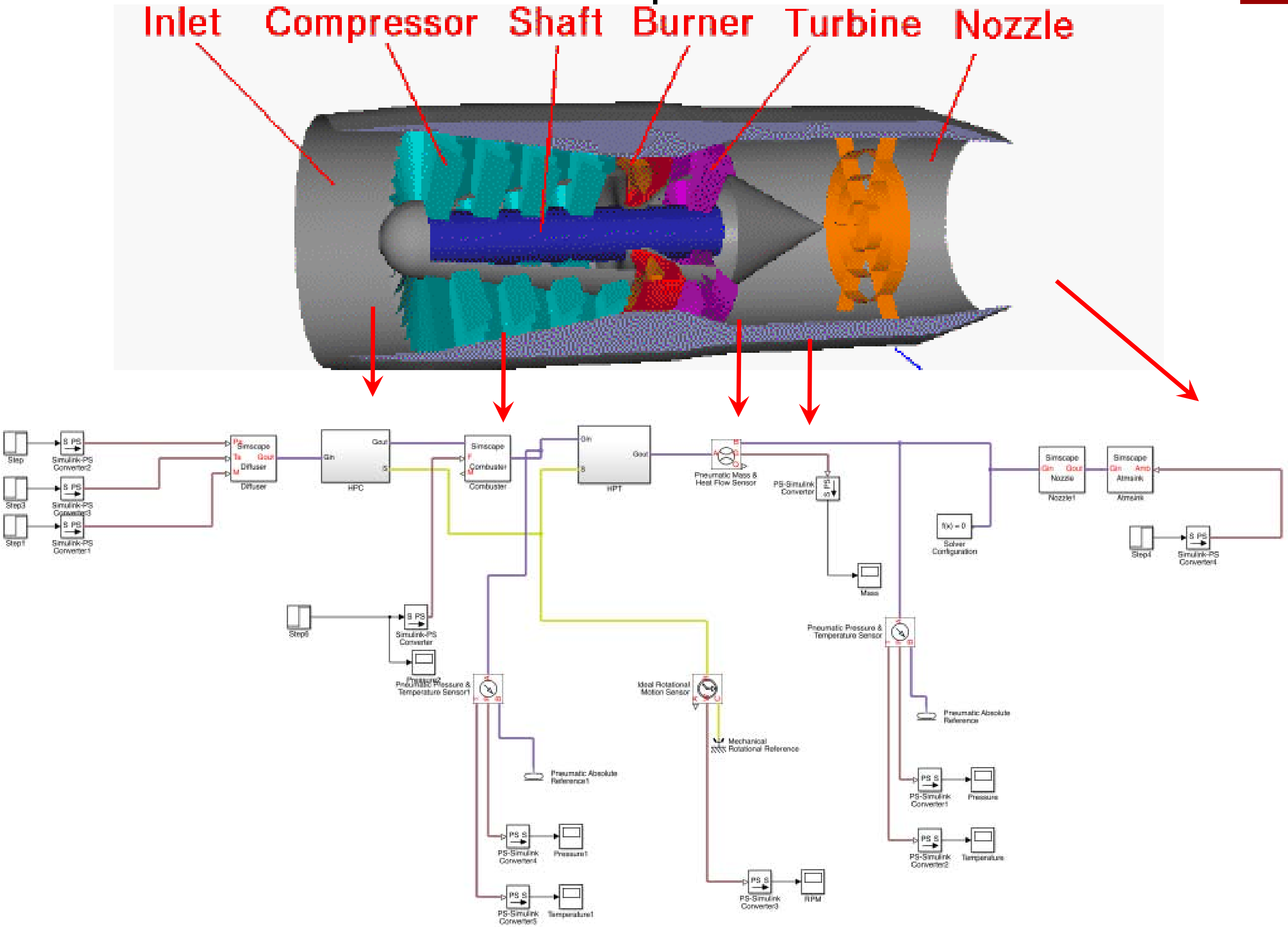


Voltage Source

Voltage Sensor

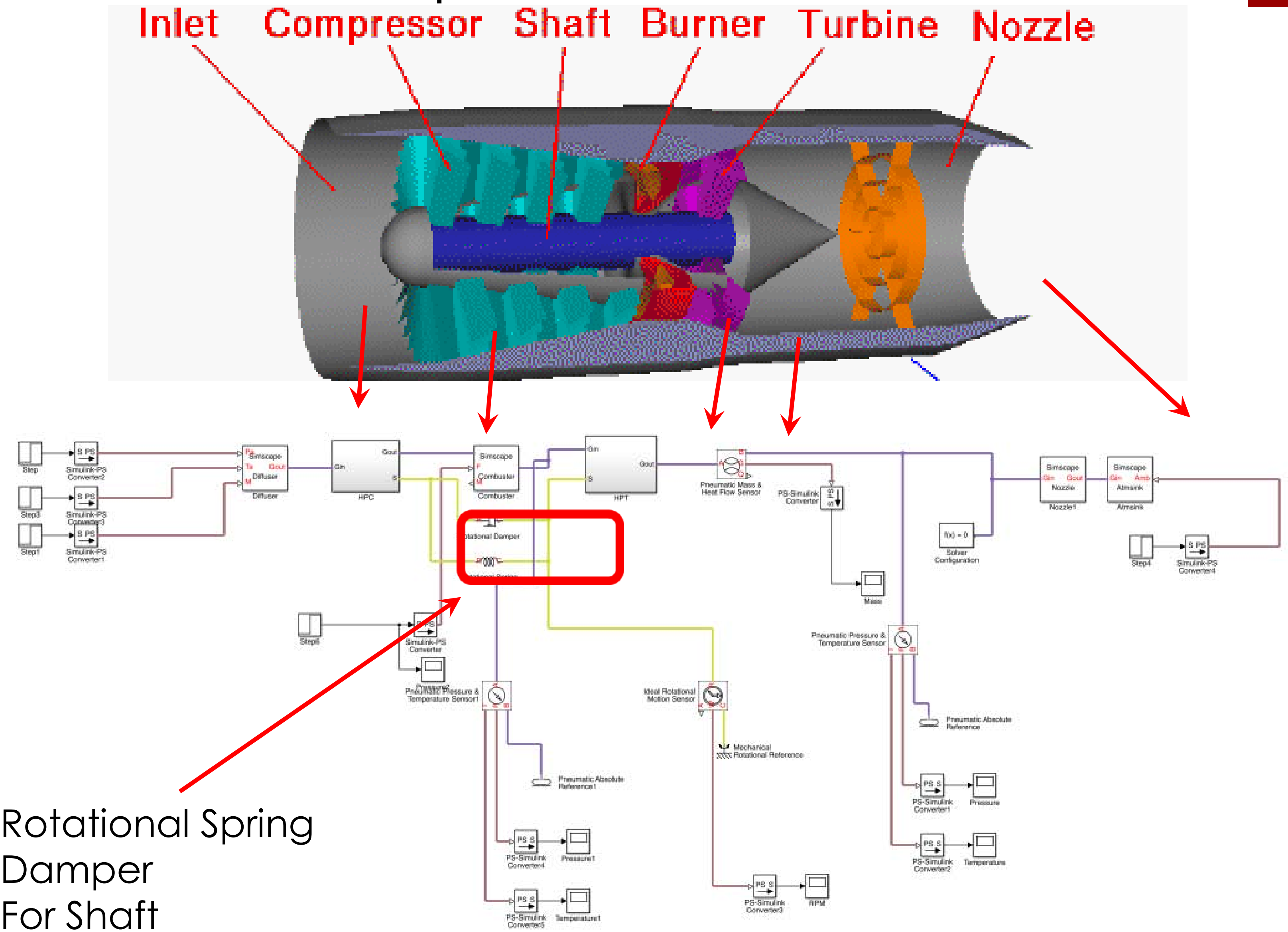
Display

Idealized Turbojet: Simscape



This image is in the public domain.

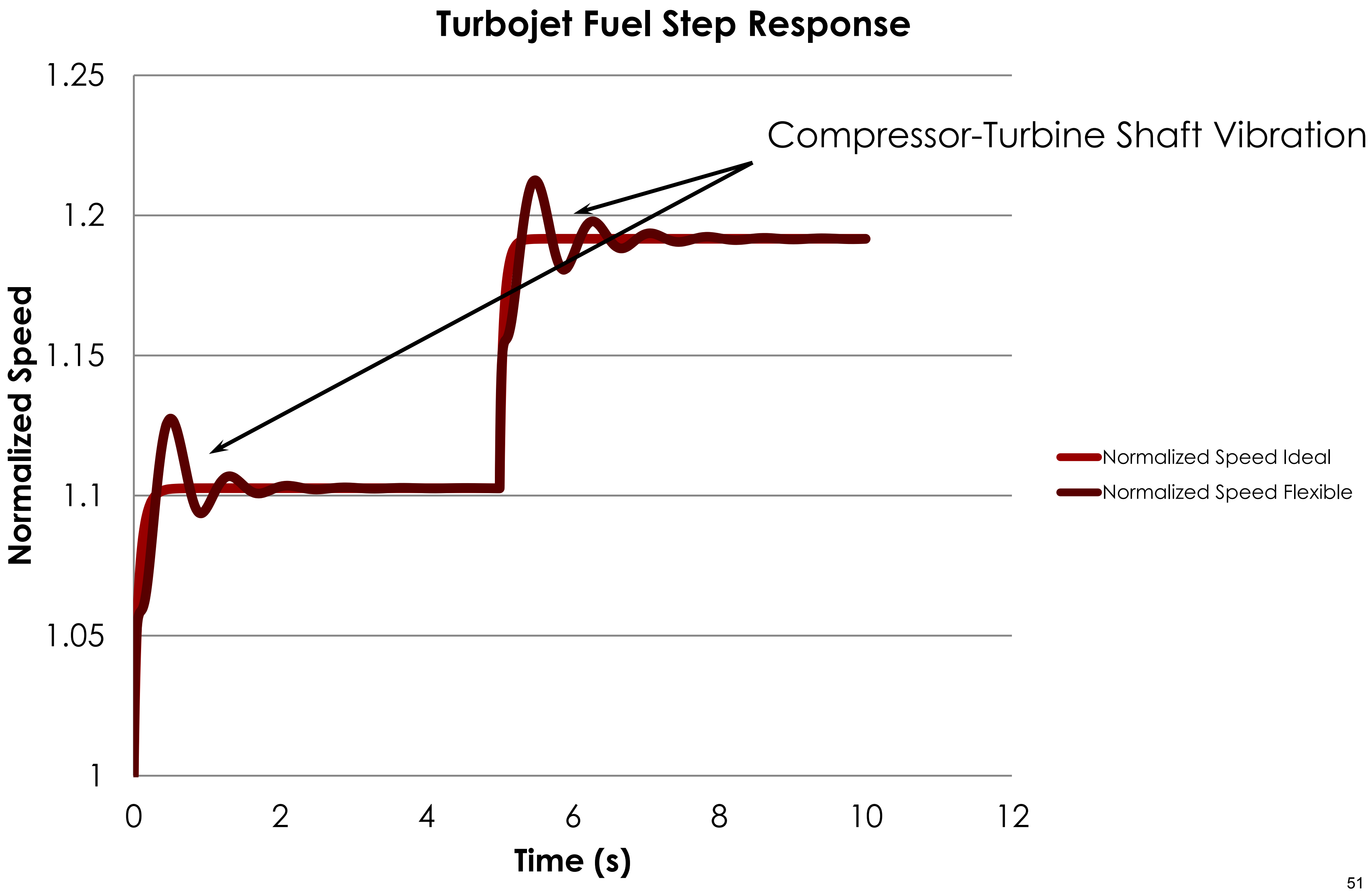
Flexible Shaft Turbojet: Simscape



Rotational Spring Damper For Shaft

This image is in the public domain.

Comparing Rigid and Flexible Engines: Simscape

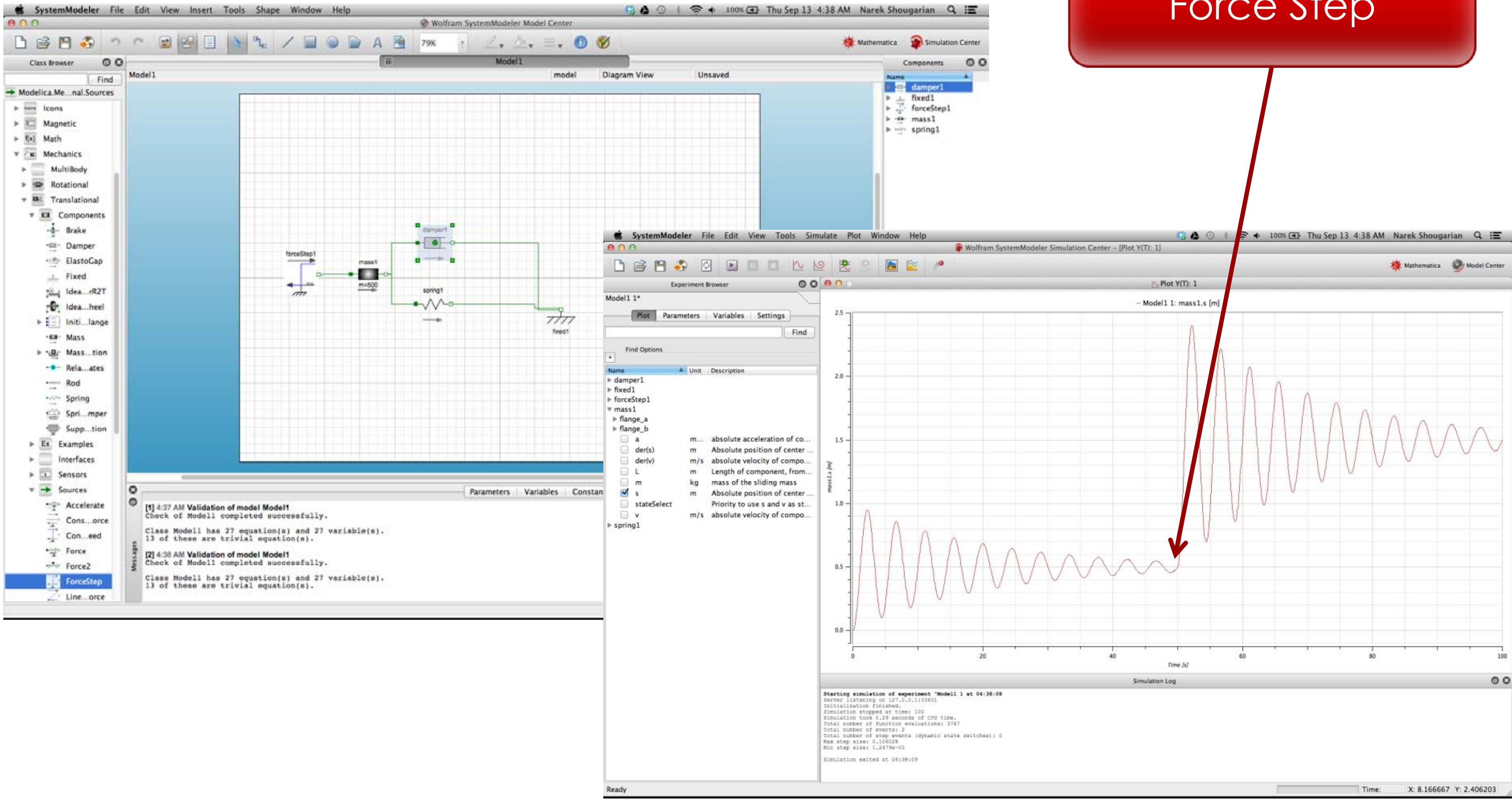


SystemModeler (Wolfram)

Simple Suspension Example

- User Friendly
- Integrated with Mathematica and Wolfram Alpha.
- Not open source

Force Step



```
[1] 4:37 AM Validation of model Model1
Check of Model1 completed successfully.

Class Model1 has 27 equation(s) and 27 variable(s).
13 of these are trivial equation(s).

[2] 4:38 AM Validation of model Model1
Check of Model1 completed successfully.

Class Model1 has 27 equation(s) and 27 variable(s).
13 of these are trivial equation(s).
```

Name	Unit	Description
▸ damper1		
▸ fixed1		
▸ forceStep1		
▾ mass1		
▸ flange_a		
▸ flange_b		
□ a	m...	absolute acceleration of co...
□ der(s)	m	Absolute position of center ...
□ der(v)	m/s	absolute velocity of compo...
□ L	m	Length of component, from...
□ m	kg	mass of the sliding mass
☑ s	m	Absolute position of center ...
□ stateSelect		Priority to use s and v as st...
□ v	m/s	absolute velocity of compo...
▸ spring1		

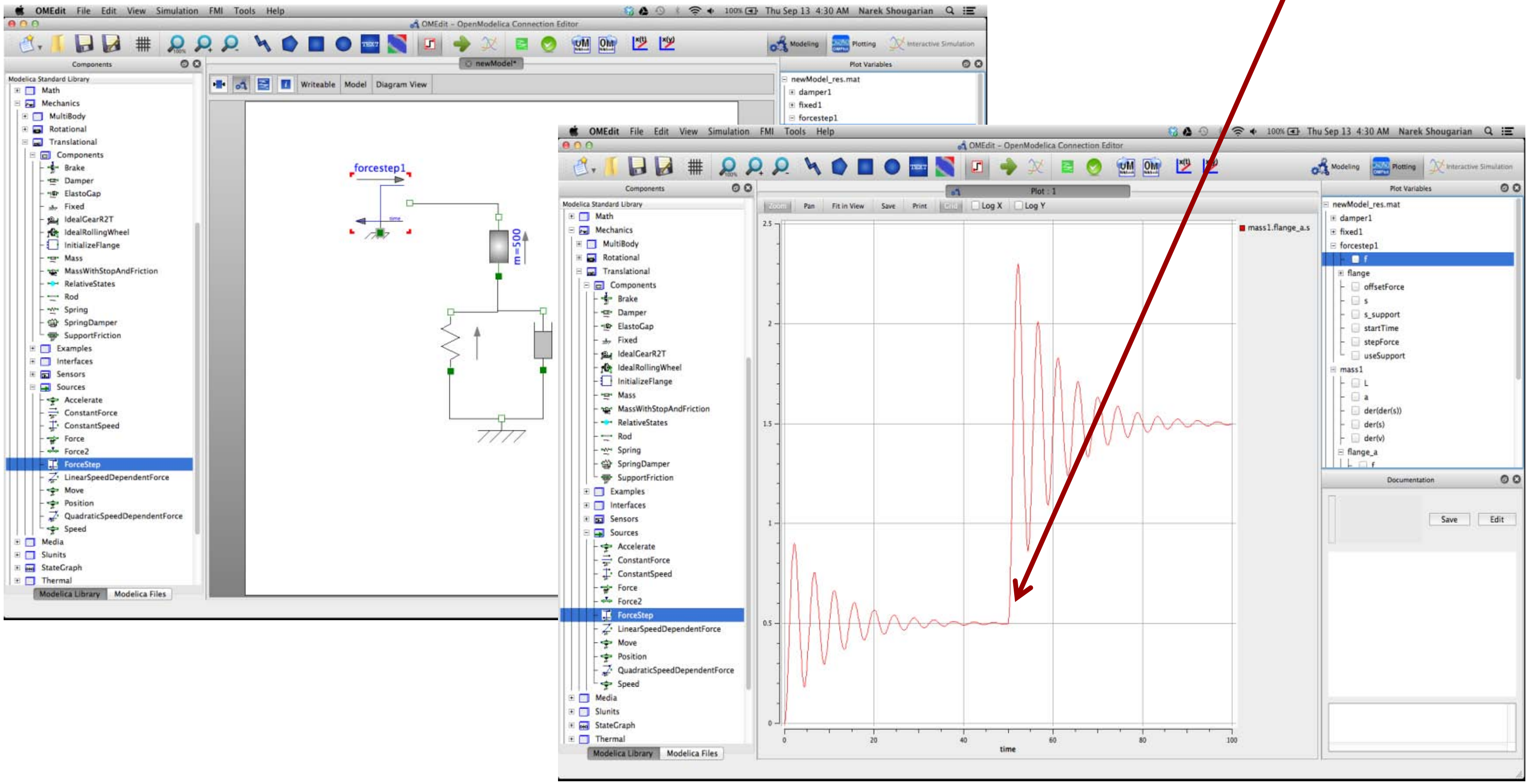
```
Starting simulation of experiment "Model1 1 at 04:38:08
Server listening on 127.0.0.1:55651
Initialization finished.
Simulation stopped at time: 100
Simulation took 0.29 seconds of CPU time.
Total number of function evaluations: 3767
Total number of events: 2
Total number of step events (dynamic state switches): 0
Max step size: 0.106028
Min step size: 1.2479e-05

Simulation exited at 04:38:09
```

OpenModelica

Simple Suspension Example

- Open Source
- Very similar to SystemModeler from Wolfram

Force Step



Modelica Demo

- Modelica Demo by Narek Shougarian

- OpenModelica Installation

  - https://www.openmodelica.org : OpenModelica for Windows,
    MAC and Linux platforms

  - https://modelica.org : Modelica and Modelica Association website. Documentation, tutorials, user uploaded libraries and publications.

Overview

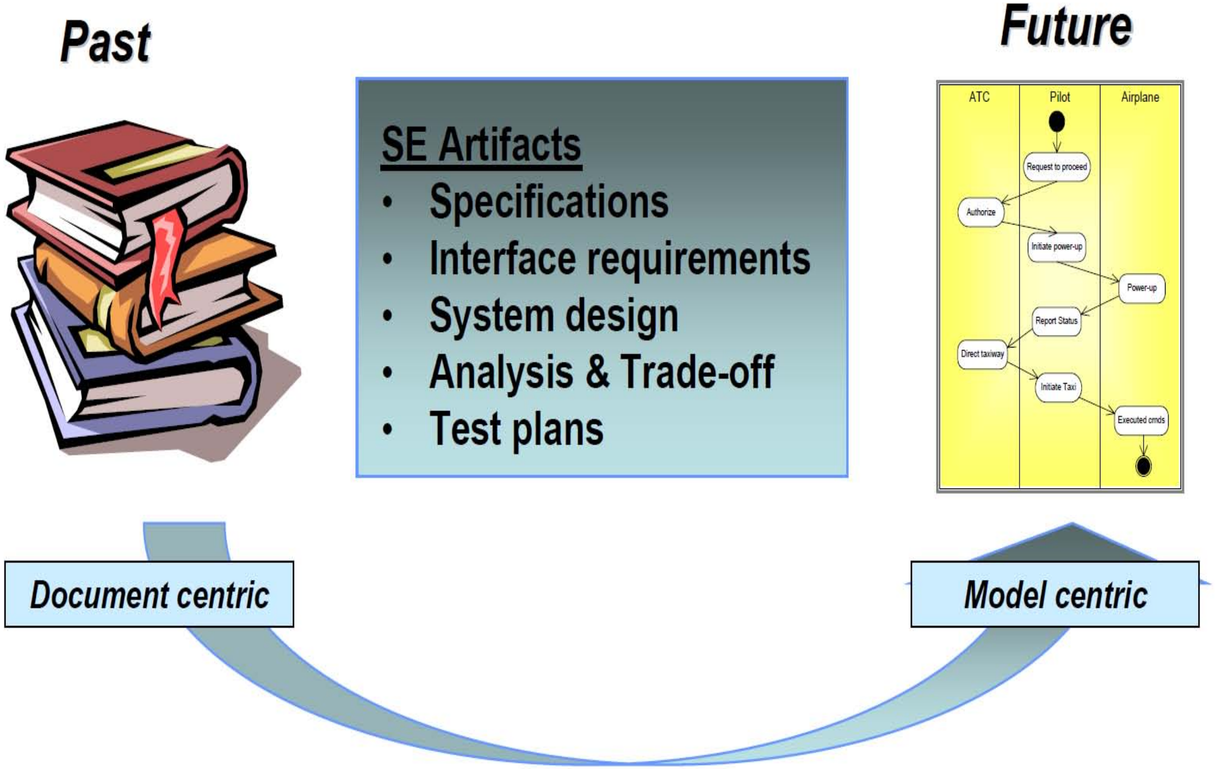
- Why Systems Modeling Languages?
  - Ontology, Semantics and Syntax
- OPM – Object Process Methodology
- SySML – Systems Modeling Language
- Modelica
- What does it mean for Systems Engineering of today and tomorrow (MBSE)?

Concept Question 5

- Which of the system modeling languages seems most useful to you?
- A – OPM
- B – SysML
- C – Modelica
- D – None of them

Answer Concept Question 5 (see supplemental files)

Model-Based Systems Engineering



Past

SE Artifacts

- Specifications
- Interface requirements
- System design
- Analysis & Trade-off
- Test plans

Future

Document centric

Model centric

Descriptive models, instead of documents, are the information storage and communication medium

Session 3 Summary

- Traditional Systems Engineering produces documents
  - E.g. Requirements Document, Interface Control Document etc…
  - Written in natural language
  - Many downsides: changes do not propagate easily, ambiguous interpretations

- Model-based Systems Engineering (MBSE)
  - Replace Documents with (executable) models
  - Need rigorous System Modeling Languages
    - Ontology, Semantics, Syntax
    - Object-Process Methodology (OPM) – Excellent for pre-Phase A
    - SysML – Widely used in some industries, 9 diagram types
    - Modelica – Declarative language, able to execute models in the time domain to simulate steady-state and transient behavior

  - Field is in transition currently …