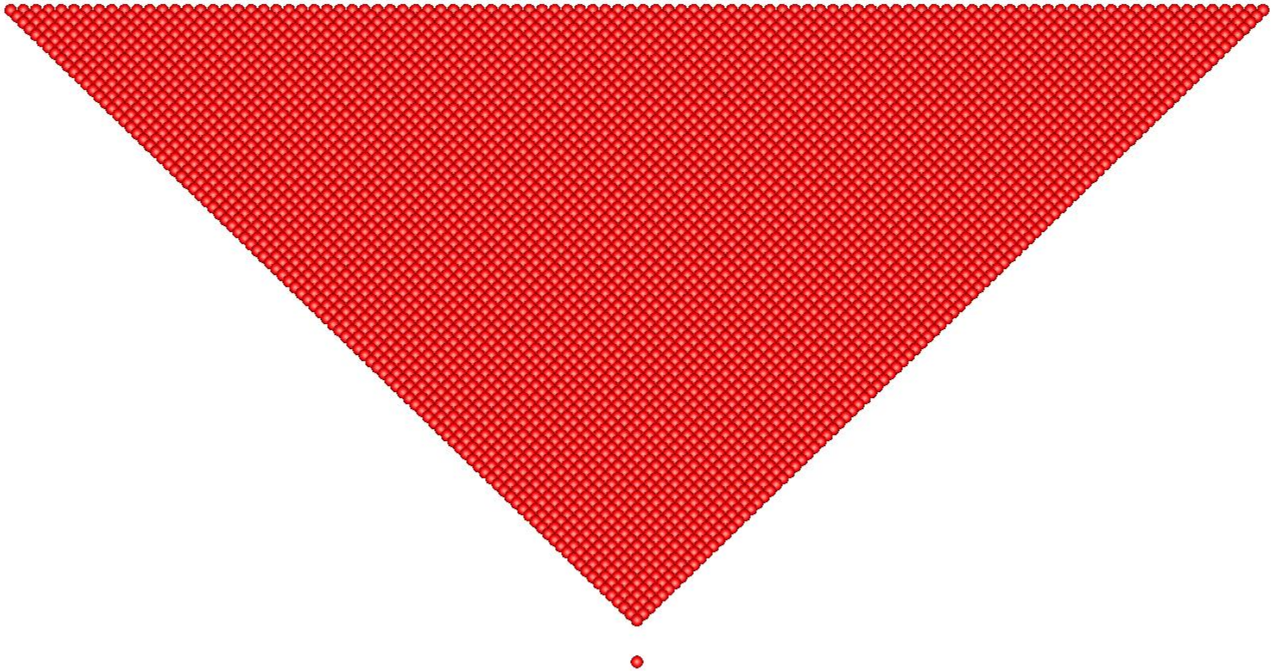
Modelica, FMI and Dymola for MBSE

Hilding Elmqvist

A model



Content

- History
- Modelica basics
- 3DExperience for Systems
- FMI
- System Engineering Model Views and Experiences

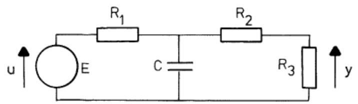
History

Dymola – Dynamic Modeling Language

- The Idea: Thursday, April 15 before Easter 1976
  - **Equations!**
- Leading to:
  - Object oriented
  - Physically oriented coupling
  - Structural analysis by graph theory
  - Computer algebra
- Boiler model coded in 8 pages
- 250 equations
- 11 systems of simultaneous equations
  - The largest 17 equations

```
model type capacitor
  cut A (Va / I) B (Vb / -I)
  main cut C [A B]
  main path P <A - B>
  local V
  parameter C
  V = Va-Vb
  C*der(V) = I
end
```

```
model Network
  submodel(resistor) R1 R2 R3
  submodel(capacitor) C
  submodel(voltage) E
  submodel Common
  input u
  output y
  connect Common to E to R1 to (C par (R2 to R3))to Common
  E.V = u
  y = R3.Va
end
```



Dymola program

- Wrote Dymola compiler in Simula language in beginning of 1978
  - Structural analysis by graph theory (Assignment, BLT)
  - Own computer algebra algorithms
- PhD Dissertation in May 1978
- Stopped working on this 1978
  - Could ONLY handle 250 equations
  - In about 128 kByte of memory on Univac-1108 computer
  - Later translated to Pascal for VAX

1992-2006

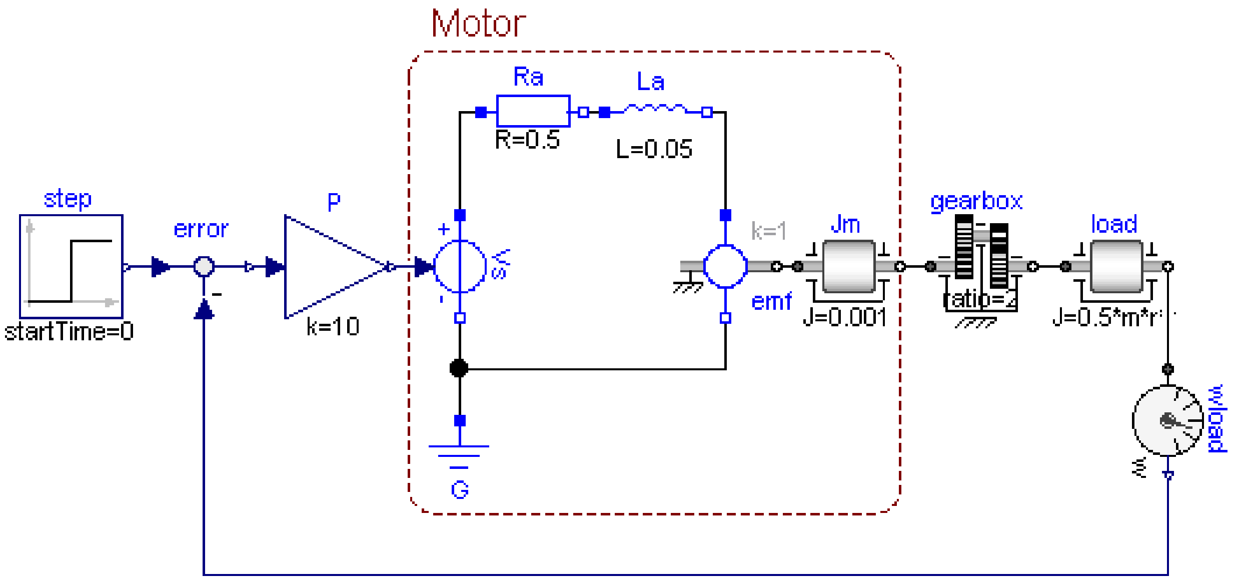
- Resumed Dymola work in 1992
  - Francois Celliers book Continuous Systems Modeling dealing with Dymola
  - Windows 3.0 got linear address space (no 640 kByte barrier)
  - Founded Dynasim AB 1992
  - Started collaborating with Martin Otter, DLR summer 1992
  - Introduced hybrid features in Dymola 1993 with Martin Otter and Francois Cellier
- Toyota started to use Dymola in 1996 for Prius development
- Started Modelica effort 1996, chairman until 1999
- Dassault Systèmes acquired Dynasim in 2006

First Modelica Design Meeting, Lund, September 1996

Alexandre Jeandel, Gaz de France **Sven Erik Mattsson**, Lund University
Martin Otter, DLR **Per Sahlin**, Brisdata/Equa **Bernt Nilsson**, Lund University
Hilding Elmqvist, Dynasim

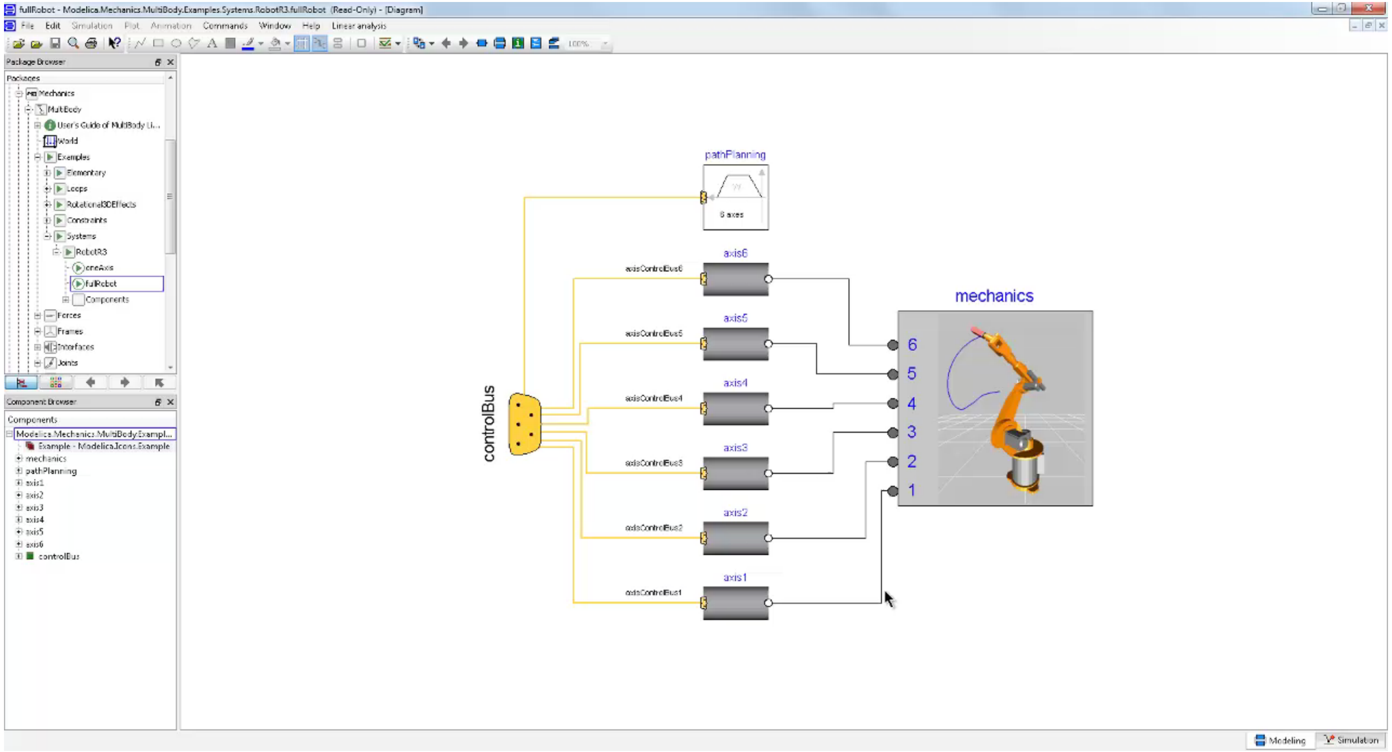
Modelica Basics

How to model this servo system?



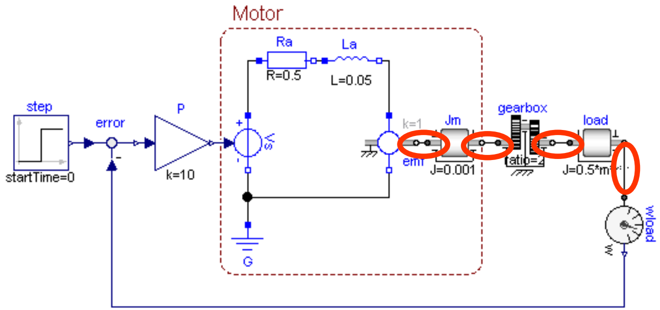
- This IS the Modelica model
- Using models from Modelica Standard Library

More Complex Model – Information Zooming



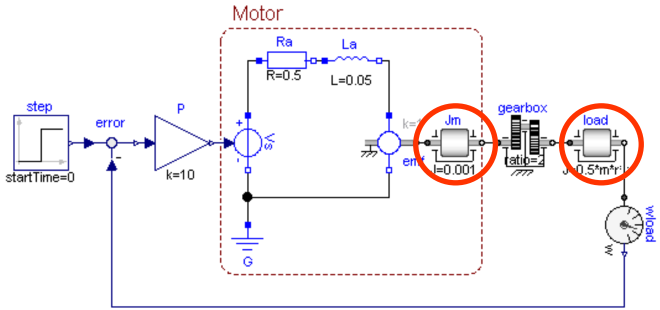
Rotational Flange

```
connector Flange "1-dim. rotational flange of a shaft"
    SI.Angle phi "Absolute rotation angle of flange";
    flow SI.Torque tau "Cut torque in the flange";
end Flange;
```



Inertia

```
model Inertia "1D-rotational component with inertia"
  Flange flange_a "Left flange of shaft";
  Flange flange_b "Right flange of shaft";
  parameter SI.Inertia J(min=0) "Moment of inertia";
  SI.AngularVelocity w "Absolute angular velocity of component";
equation
  flange_a.phi = flange_b.phi;
  w = der(flange_a.phi);
  J*der(w) = flange_a.tau + flange_b.tau;
end Inertia;
```

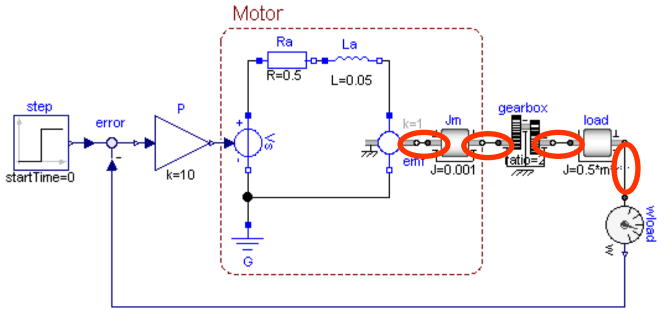


MotorDrive model

```
model MotorDrive
  parameter Modelica.SIunits.Radius r=0.5 "Radius of load";
  parameter Modelica.SIunits.Mass m=80 "Mass of load";
  Modelica.Mechanics.Rotational.Components.IdealGear gearbox(ratio=2);
  Modelica.Mechanics.Rotational.Components.Inertia load(J=0.5*m*r*r);
  Modelica.Mechanics.Rotational.Sensors.SpeedSensor wload;
  Modelica.Blocks.Math.Feedback error;
  Modelica.Blocks.Math.Gain P(k=10);
  Modelica.Electrical.Analog.Sources.SignalVoltage Vs;
  Modelica.Electrical.Analog.Basic.Ground G;
  Modelica.Electrical.Analog.Basic.Resistor Ra(R=0.5);
  Modelica.Electrical.Analog.Basic.Inductor La(L=0.05);
  Modelica.Electrical.Analog.Basic.EMF emf(k=1);
  Modelica.Mechanics.Rotational.Components.Inertia Jm(J=0.001);
  Modelica.Blocks.Sources.Step step;
…
end MotorDrive;
```

MotorDrive model (continued)

```
model MotorDrive
…
equation
  connect(gearbox.flange_b, load.flange_a);        ←
  connect(load.flange_b, wload.flange);            ←
  connect(error.y, P.u);
  connect(wload.w, error.u2);
  connect(Ra.n, La.p);
  connect(La.n, emf.p);
  connect(emf.flange, Jm.flange_a);                ←
  connect(Ra.p, Vs.p);
  connect(Vs.n, emf.n);
  connect(G.p, Vs.n);
  connect(P.y, Vs.v);
  connect(Jm.flange_b, gearbox.flange_a);          ←
  connect(step.y, error.u1);
end MotorDrive;
```



Inside Dymola

Equations – incidence matrix

Components equations

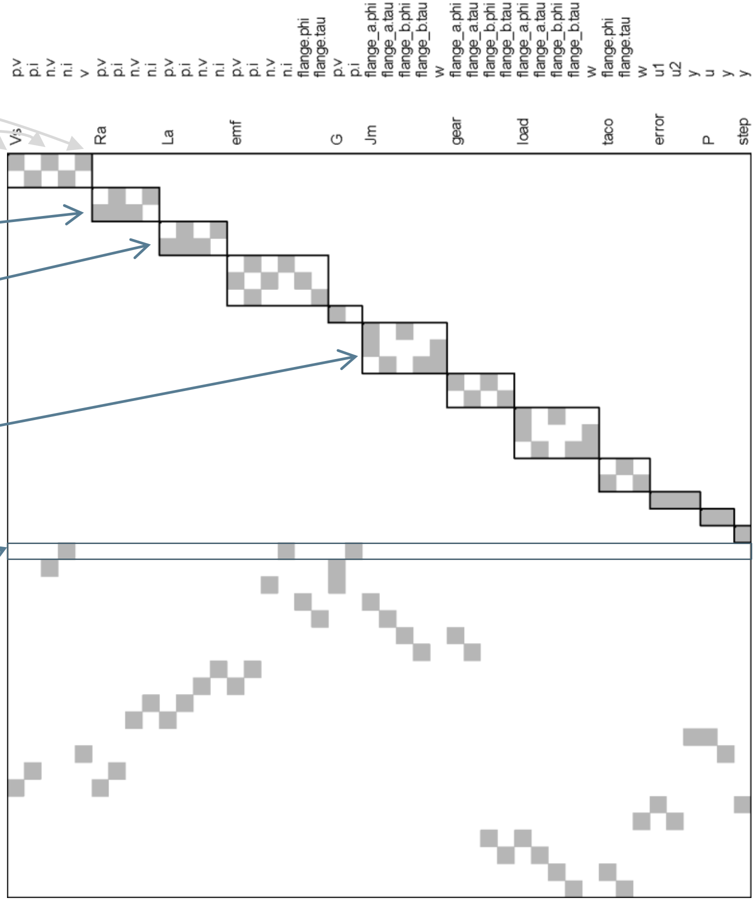
```
  // SignalVoltage Vs
Vs.v = Vs.p.v-Vs.n.v;
0 = Vs.p.i+Vs.n.i;

  // Resistor Ra
0 = Ra.p.i+Ra.n.i;
Ra.R*Ra.p.i = Ra.p.v-Ra.n.v;

  // Inductor La
0 = La.p.i+La.n.i;
La.L*der(La.p.i) = La.p.v-La.n.v;
…
  // Inertia Jm
Jm.flange_a.phi = Jm.flange_b.phi;
Jm.w = der(Jm.flange_a.phi);
Jm.J*der(Jm.w) = Jm.flange_a.tau+Jm.flange_b.tau;
…
```

Connection equations

```
  // model MotorDrive
G.p.i+Vs.n.i+emf.n.i = 0.0;
Vs.n.v = G.p.v;
…
```



Structural analysis and computer algebra

- Angles of inertias load and Jm are constrained by the gearbox
  - there is only one degree-of-freedom
  - the DAE is of high index.
- The index reduction algorithm needs to differentiate certain equations
- Equations are sorted to create an algorithm for sequential execution
  - Known: parameters and states
  - Unknown: derivatives and outputs
- There might be mutual dependencies between equations (algebraic loops)
  - Correspond to blocks in the incidence matrix
- After sorting, i.e. permutation of rows and columns, the incidence matrix has Block Lower Triangular form (BLT)
- Equations solved using computer algebra
  - The expressions are represented as trees and algebraic transformation rules are applied.
- For linear blocks, linear algebra routines are used.
- For non-linear blocks, a Newton-Raphson solver is utilized.

Solved Equations

error.u1 = step.offset+(if time < step.startTime then 0 else step.height);

error.y = error.u1-load.w;

Vs.p.v = P.k*error.y;

Ra.R*La.p.i = Vs.p.v-**Ra.n.v**;

Jm.w = gear.ratio*load.w;

emf.k*Jm.w = **La.n.v**;

La.L***der(La.p.i)** = Ra.n.v-La.n.v;

emf.flange.tau = -emf.k*La.p.i;

// System of 4 simultaneous equations

der(Jm.w) = gear.ratio*der(load.w);

Jm.J*der(Jm.w) = **Jm.flange_b.tau**-emf.flange.tau;

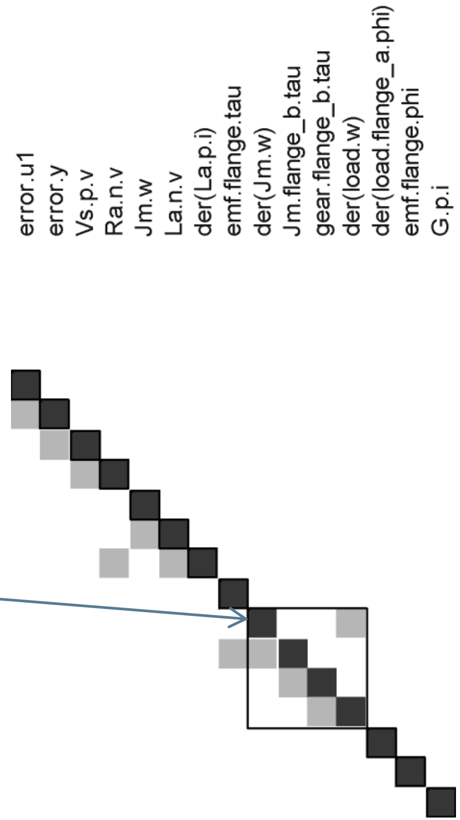
0 = **gear.flange_b.tau**-gear.ratio*Jm.flange_b.tau;

load.J***der(load.w)** = -gear.flange_b.tau;

der(load.flange_a.phi) = load.w;

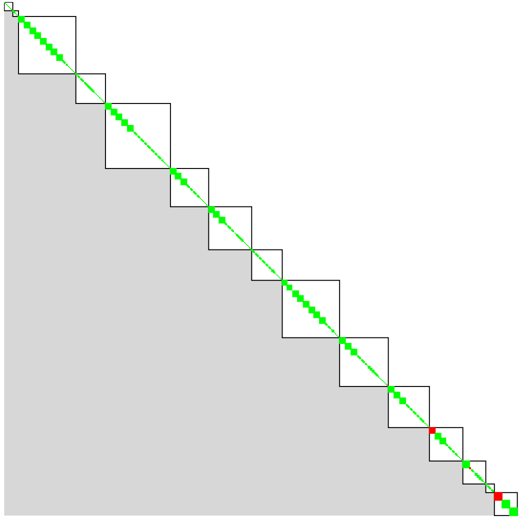
emf.flange.phi = gear.ratio*load.flange_a.phi;

G.p.i+La.p.i = La.p.i;

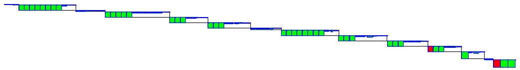


Parallelization for many cores

- BLT gives one possible block execution order
- Utilize zeros below diagonal
- Compress vertically

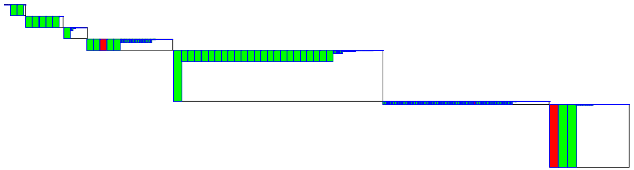


BLT structure



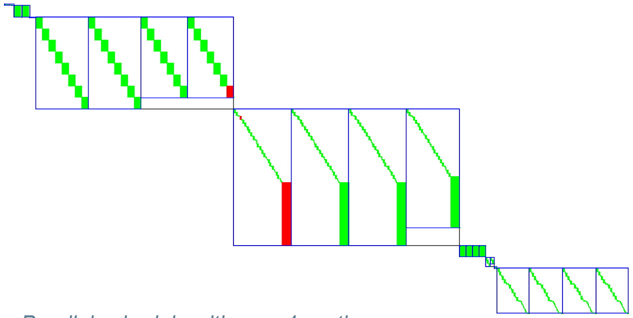
Initial parallel schedule

Parallelization



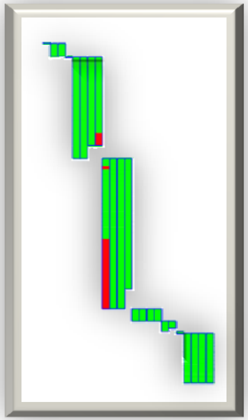
Parallel schedule with cost

- speedUpFactor = **7.0**
- numberOfLayers = 15
- numberOfCores = 325



Parallel schedule with max 4 sections

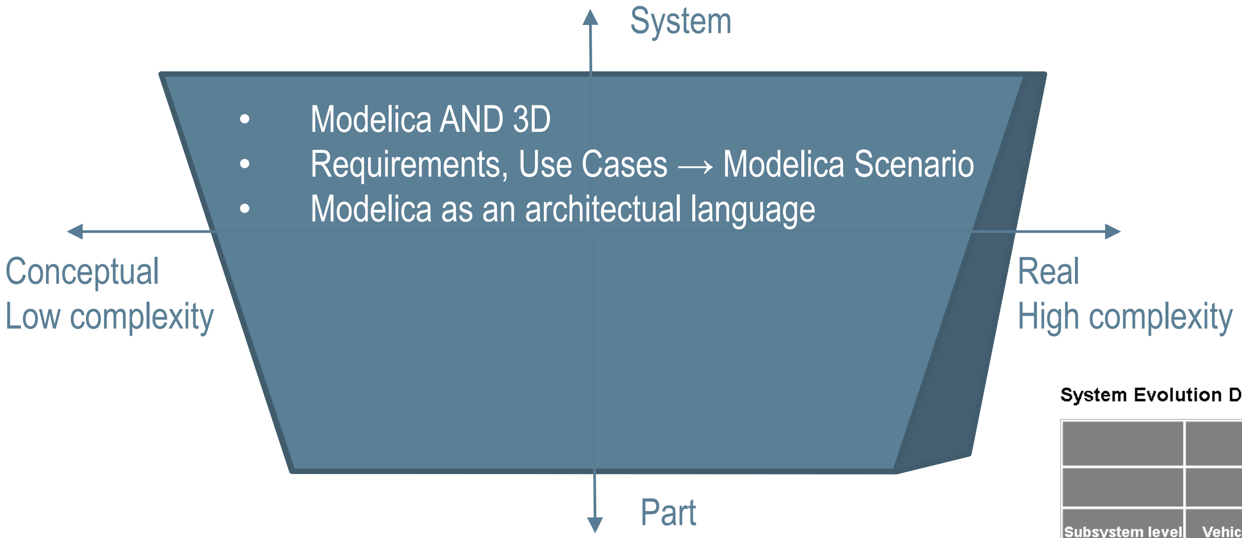
- speedUpFactor = **3.7**
- numberOfLayers = 6
- numberOfCores = 4



Gantt chart for 4 section schedule

3DExperience Platform

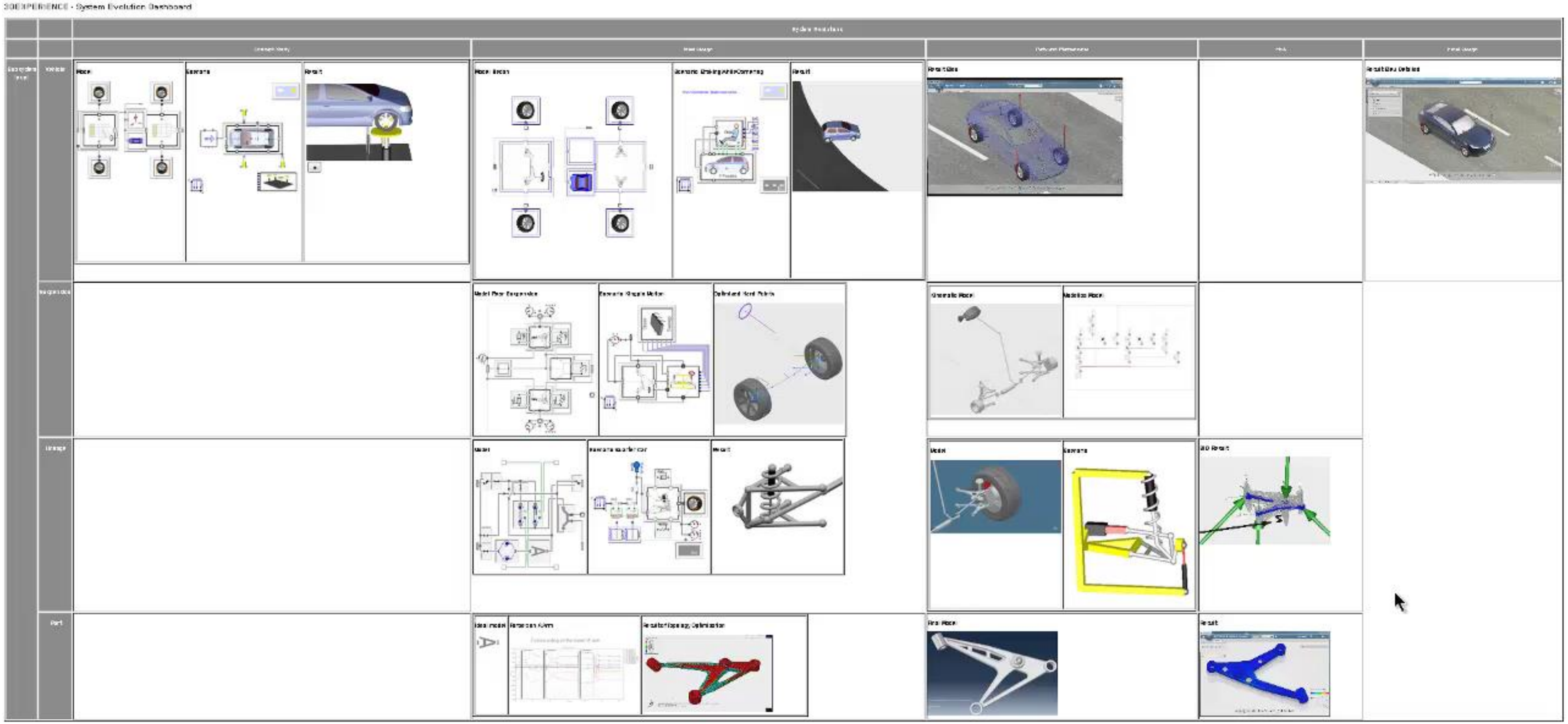
Modelica Based Systems Engineering



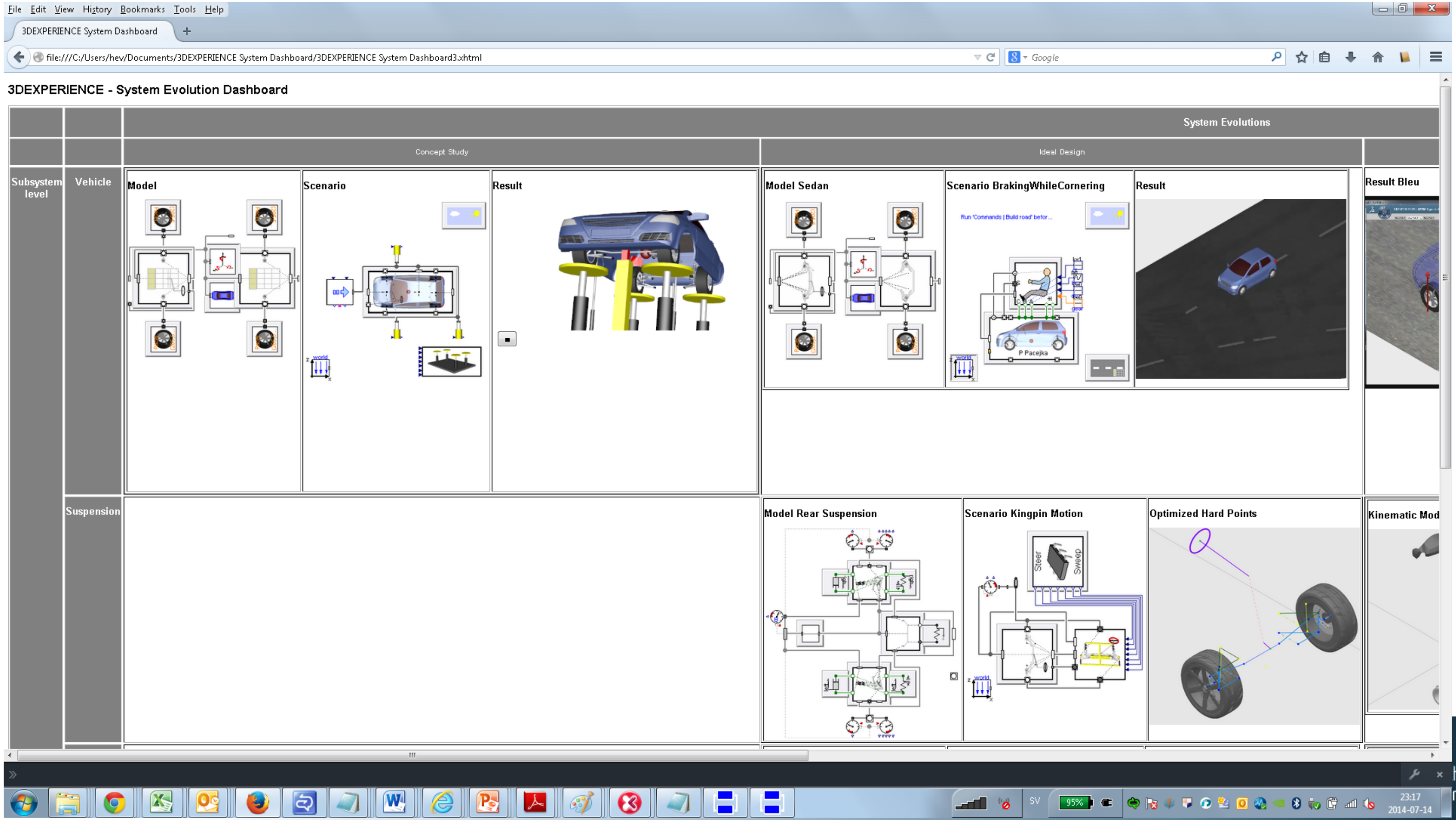
System Evolution Dashboard

		System Evolutions				
		Concept Study	Ideal Design	Parts and Mechanisms	FEA	Final Design
Subsystem level	Vehicle	Model Scenario Result				
	Suspension					
	Linkage					
	Part					

System Evolution Dashboard (Mock-up)



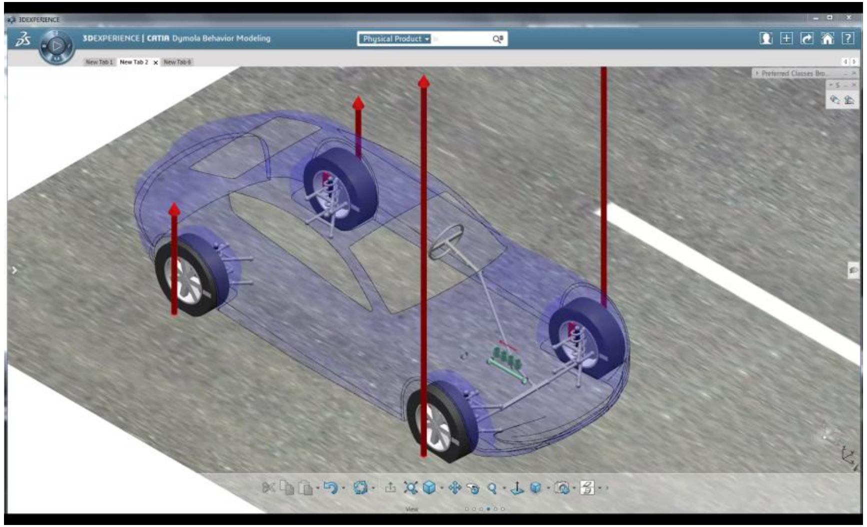
System Evolution Dashboard

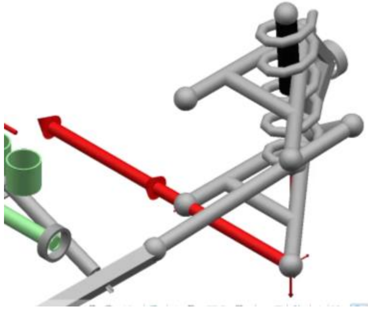


he right questions
nge the world.

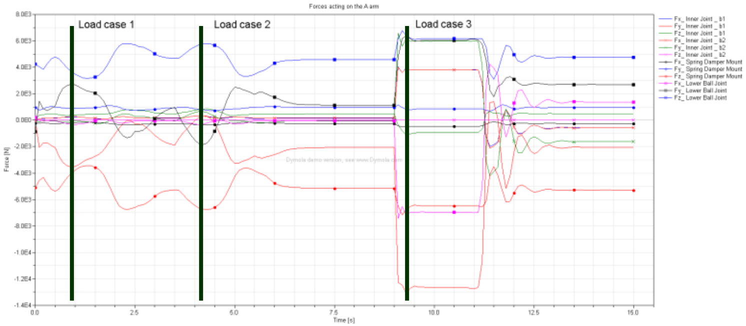
Double Lane Change Maneuver and Braking

- Record forces on A-arm
- Define load cases



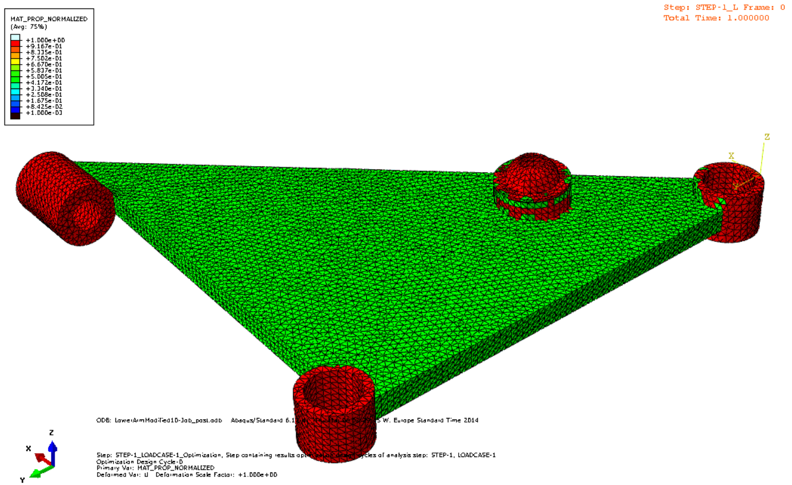


Forces acting on the lower 'A' arm



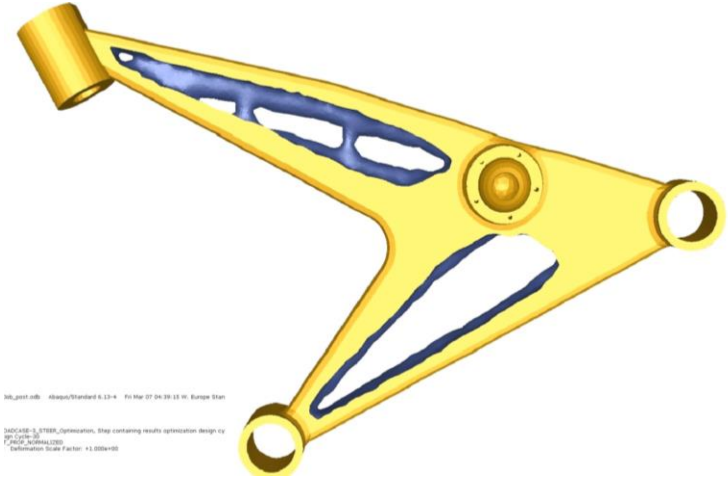
Part Design

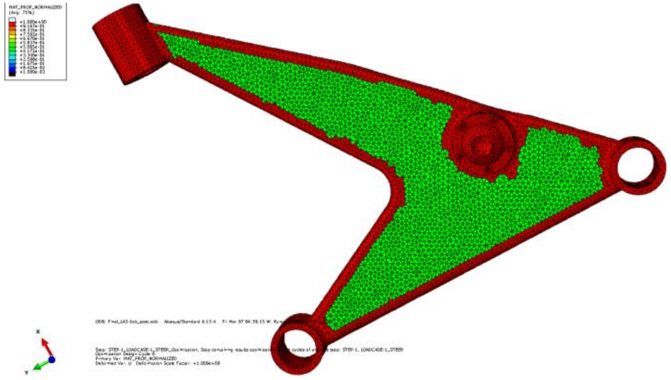
- Use Topology Optimization for A-arm
- With hard points, design area and recorded forces

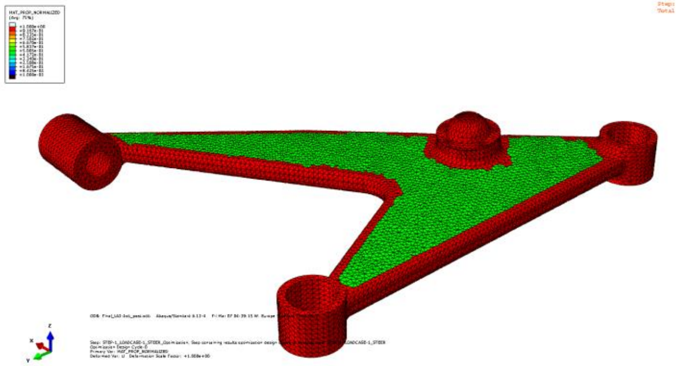


Part Design II

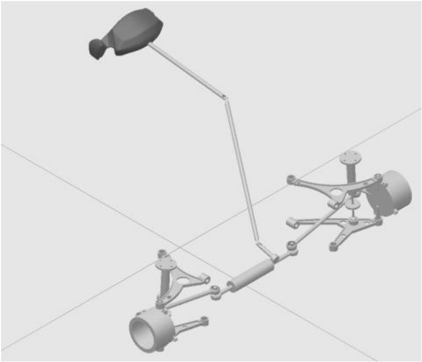
- New Design space
- Refine part design





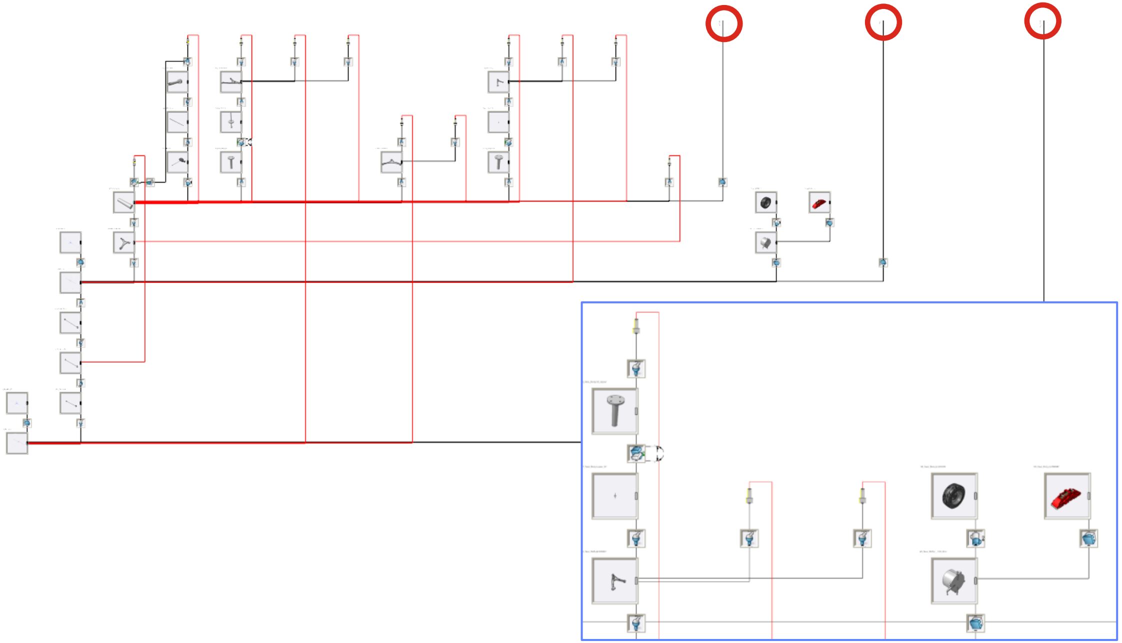


Kinematics and Modelica

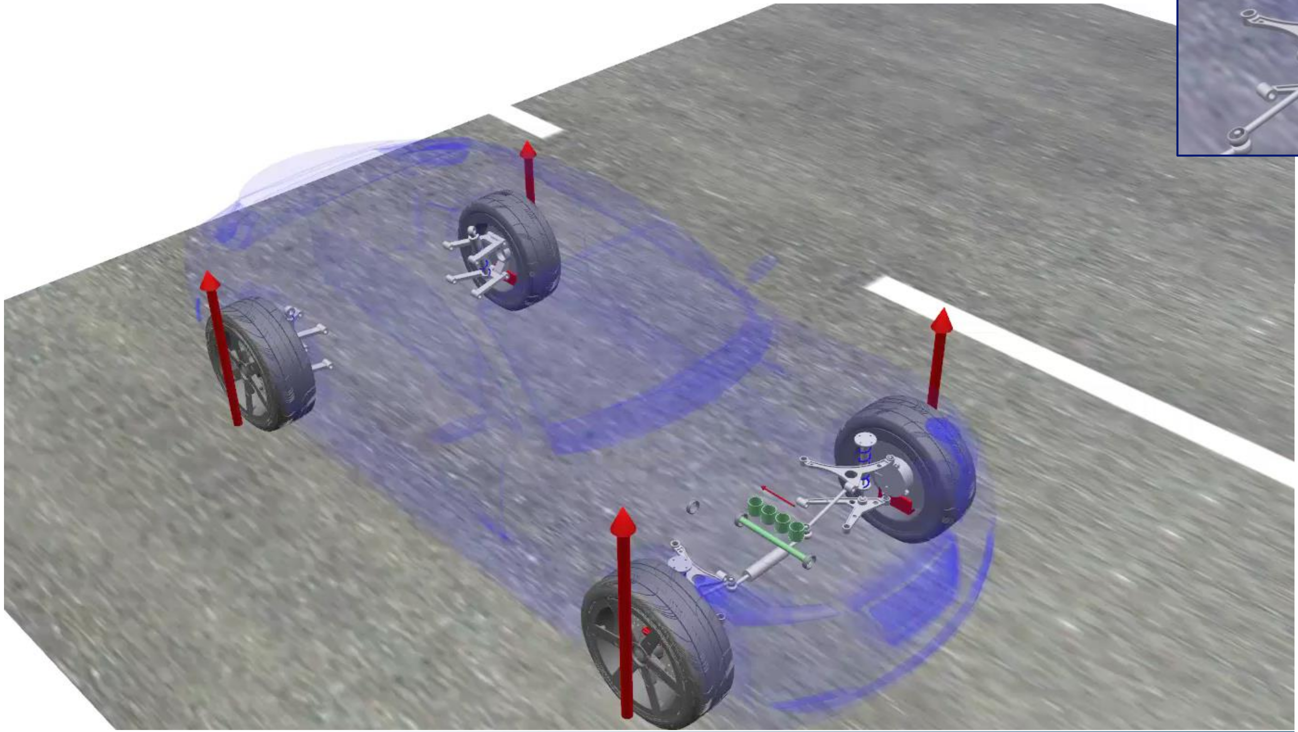


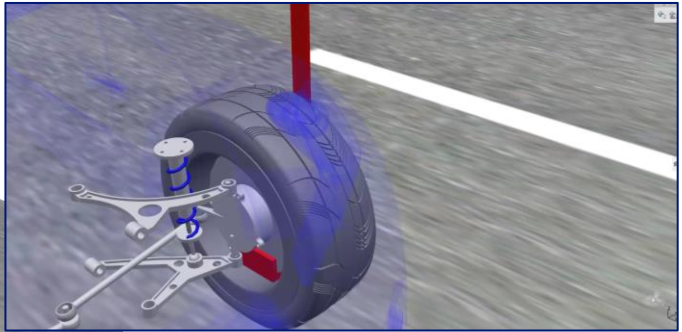
CATIA Kinematics model

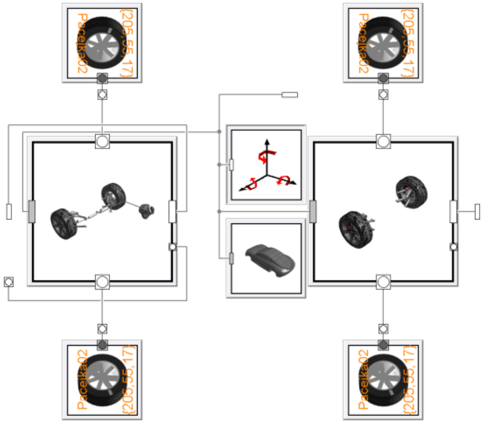
Modelica representation



Redeclared suspensions

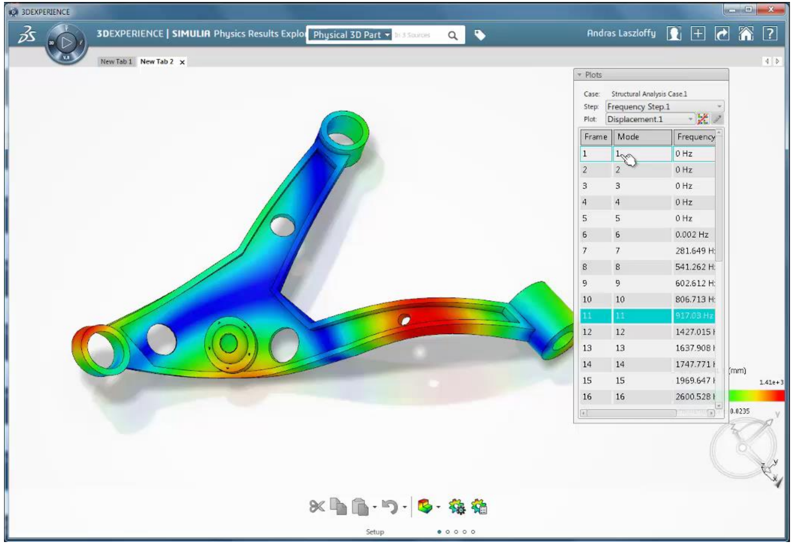


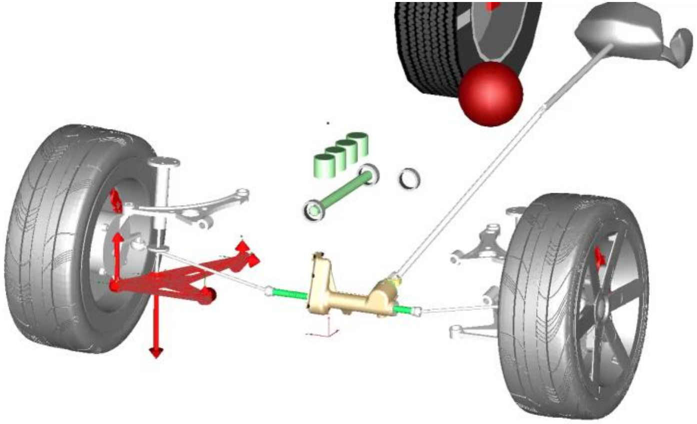




Flexible Bodies in System analysis

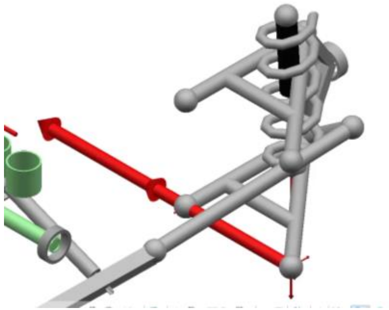
- Model reduction - Modes
- Data in SID format used in Modelica FlexibleBodies library



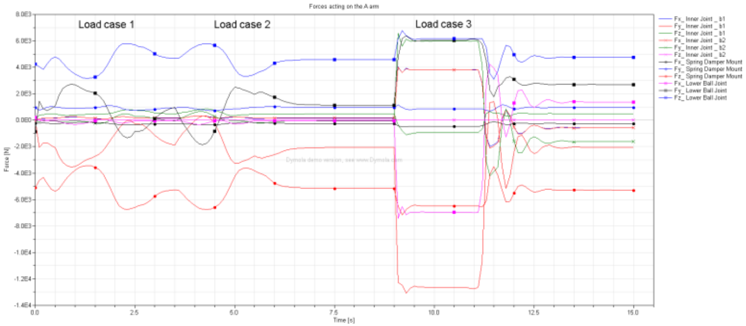


FEA of Parts

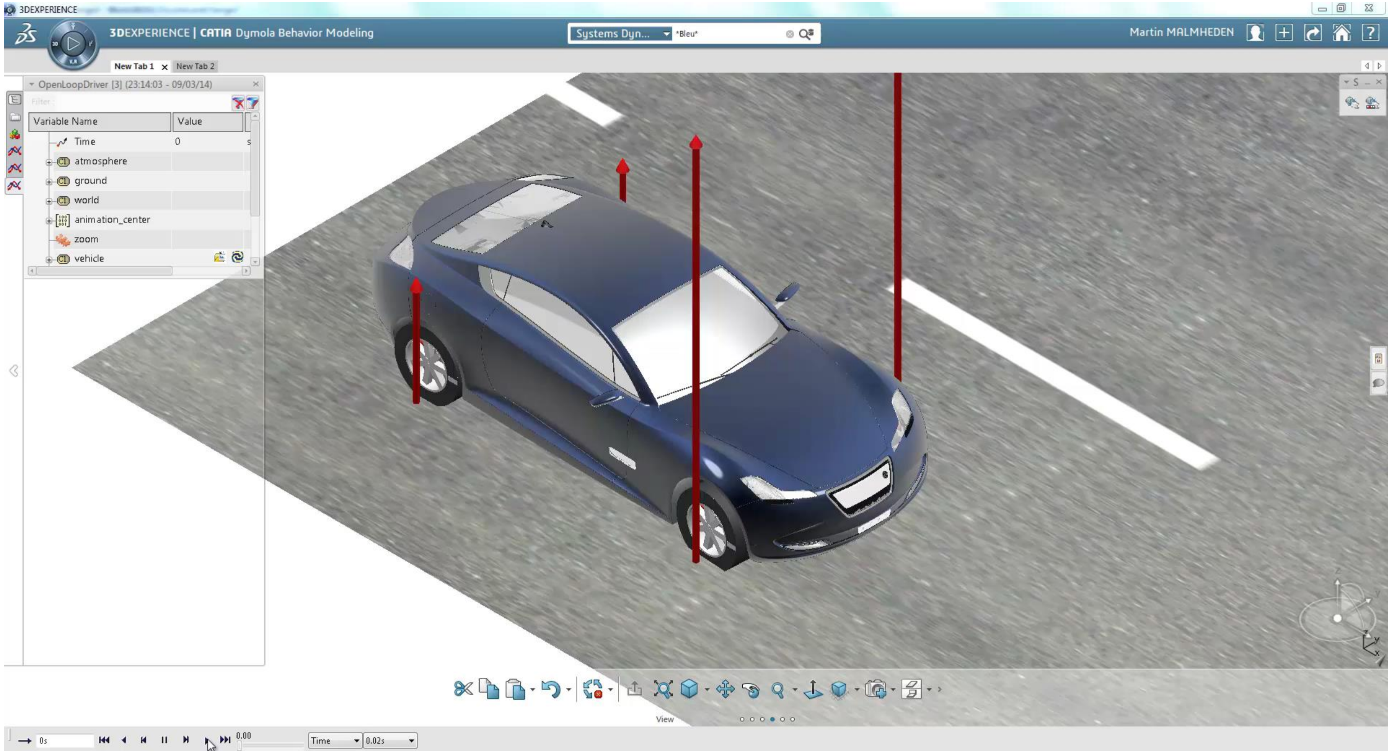
- Record forces on A-arm
  - Double Lane Change Maneuver and Braking
- Apply these as external loads for FE analysis



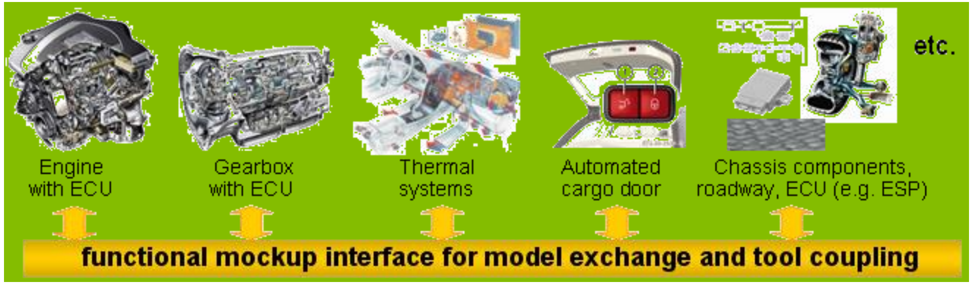
Forces acting on the lower 'A' arm



The car Bleu

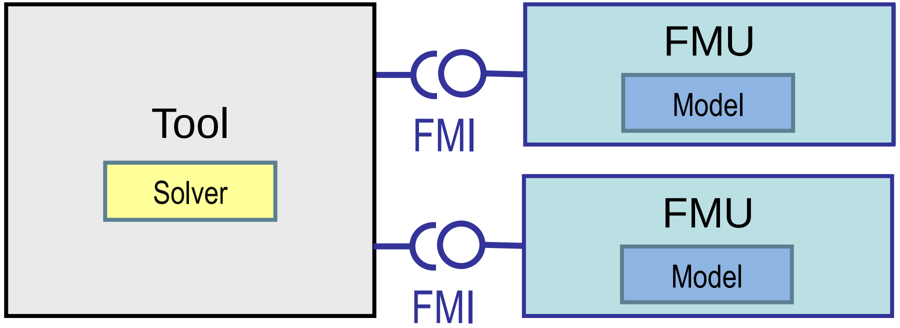

3DEXPERIENCE
3DEXPERIENCE | CATIA Dymola Behavior Modeling
Systems Dyn...
Bleu
Martin MALMHEDEN
New Tab 1
New Tab 2
OpenLoopDriver [3] (23:14:03 - 09/03/14)
Variable Name
Value
Time
0
atmosphere
ground
world
animation_center
zoom
vehicle
View
Time
0.02s

FMI – Functional Mockup Interface

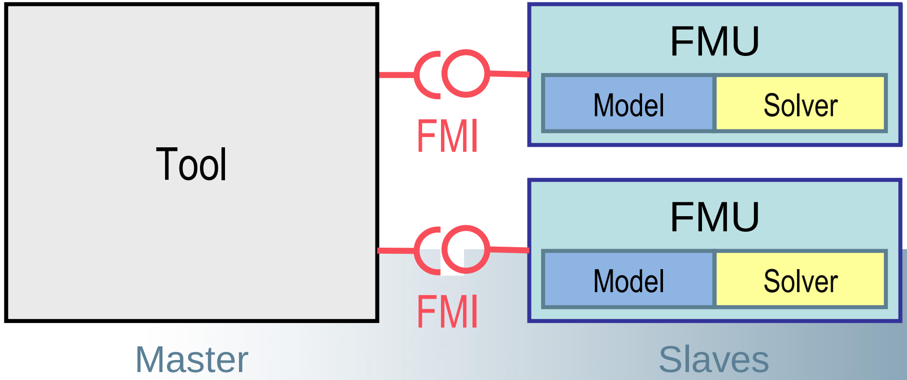


FMI – Main Design Idea

▷ FMI for Model Exchange



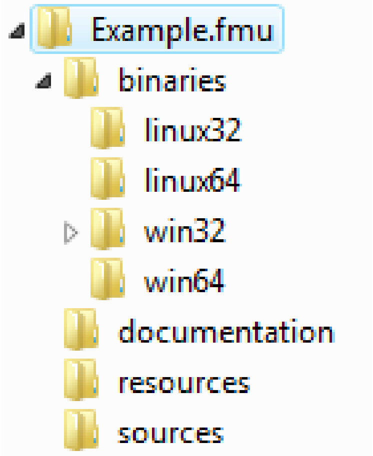
▷ FMI for Co-Simulation



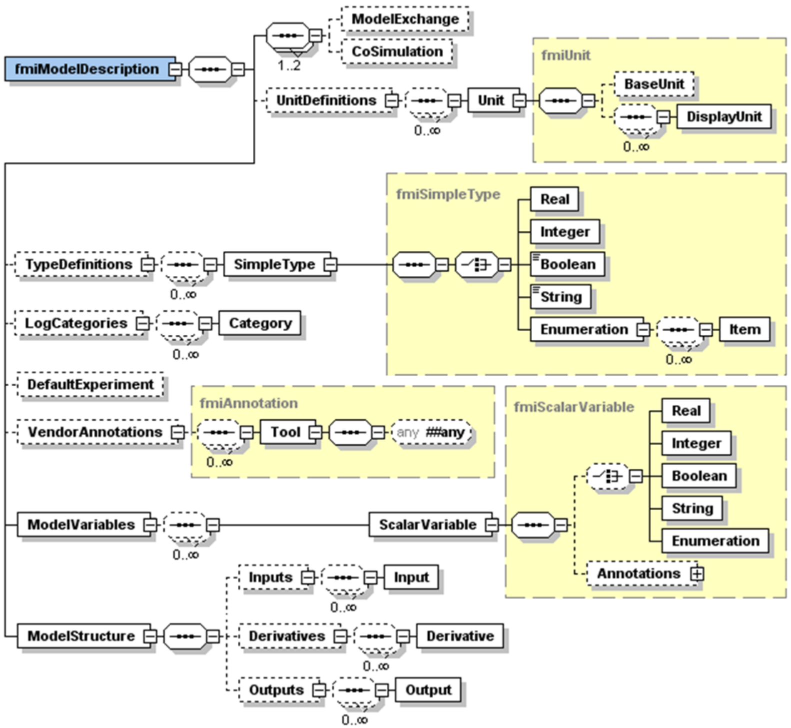
FMI Design

Packaging

- A component which implements the interface is called a *Functional Mockup Unit (FMU)*
- Separation of:
  - Description of interface data (XML file)
  - Functionality (API in C)
- An FMU is a zipped file (*.fmu) containing:
  - modelDescription.xml
  - Implementation in source and/or binary form
  - Additional data and functionality
- One FMU can contain implementations of both interfaces



XML Model Description



- *Implementation and capability flags*
- *Definition of units*

- *Definition of variable types*

- *Variables and their attributes*

- *Dependency information*

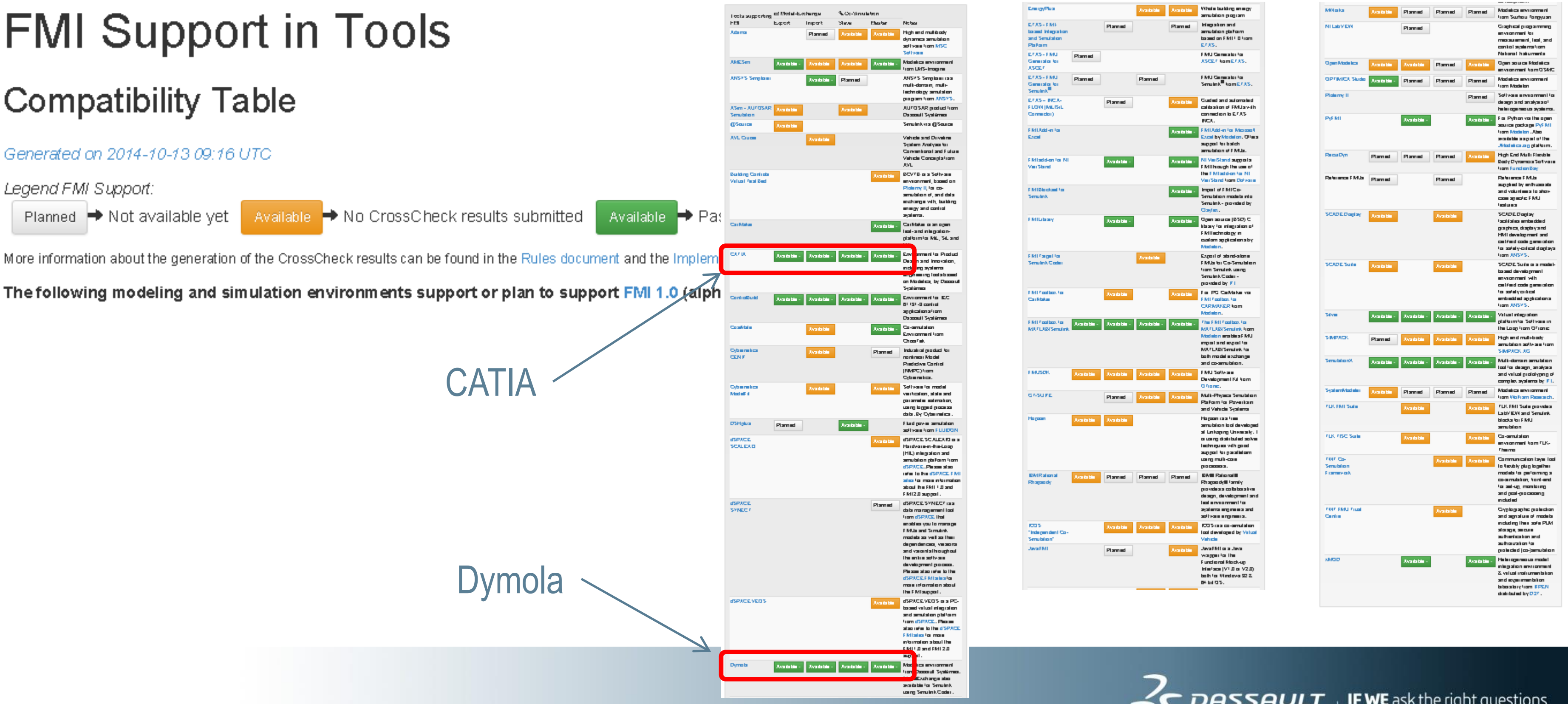
Example

```
...
<ModelVariables>
  <ScalarVariable
    name="torque"
    valueReference="335544320"
    description="Torque in flange"
    causality="output">
    <Real
      declaredType=
       "Modelica.Blocks.Interfaces.RealOutput"
      unit="N.m"/>
     ...
</ModelVariables>
<ModelStructure>
  <Inputs>
    <Input name="phi"/>
    <Input name="w" derivative="1"/>
  </Inputs>
  <Derivatives>
    <Derivative
      name="der(inertia.phi)"
      state="inertia.phi"
      stateDependencies="2"
      inputDependencies=""/>
    <Derivative
      name="der(inertia.w)"
      state="inertia.w"/>
  </Derivatives>
  <Outputs>
    <Output name="torque"
     inputDependencies="1 2"
     inputFactorKinds="fixed fixed"/>
  </Outputs>
</ModelStructure>
</fmiModelDescription>
```

C-interface

- Instantiation:
```
fmiComponent fmiInstantiateModel(fmiString instanceName, ...)
fmiComponent fmiInstantiateSlave(fmiString instanceName, ...)
```
  - Returns an instance of the FMU. Returned `fmiComponent` is an argument of the other interface functions.
- Functions for initialization, termination, destruction
- Support of real, integer, boolean, and string inputs, outputs, parameters
- `Set` and `Get` functions for each type:
```
fmiStatus fmiSetReal    (fmiComponent c,
                         const fmiValueReference vr[], size_t nvr,
                         const fmiReal value[])
fmiStatus fmiSetInteger(fmiComponent c,
                         const fmiValueReference vr[], size_t nvr,
                         const fmiInteger value[])
```
- Identification by `valueReference`, defined in the XML description file for each variable

FMI 1.0 Tool Compatibility Table



FMI 2.0
Tool Compatibility Table

https://www.fmi-standard.org/tools_FMI_2.0

Dymola

FMI Support in Tools

Compatibility Table

Generated on 2015-04-24 16:19 UTC

The following modeling and simulation environments support or plan to support FMI (alphabetical list):

Legend
- Planned → Not available yet
- Available → No CrossCheck results submitted
- Available 12 → Passed CrossCheck, 12 FMUs exported or imported, click for results

More information about the generation of the CrossCheck results can be found in the Rules document and the Implementation notes.

Total numbers | FMI_2.0 | ⇄ Model Exchange | CoSimulation

Tools supporting FMI	FMI Version	Model Exchange: Export	Model Exchange: Import	CoSimulation: Slave	CoSimulation: Master	Notes
Adams	FMI_2.0	Planned	Planned	Available	Available	High end multibody dynamics simulation software from MSC Software
Amesim	FMI_2.0			Planned	Available 37	Integrated simulation platform for the analysis of multi-domain mechatronics systems by Siemens PLM Software
ControlBuild	FMI_2.0			Available 14	Available	Environment for IEC 61131-3 control applications from Dassault Systèmes
DS - FMU Export from Simulink	FMI_2.0	Available		Available		Simulink Coder Target developed by Dassault Systèmes for export of FMUs from Simulink.
DS - FMU Import into Simulink	FMI_2.0				Planned	FMU import into Simulink developed by Dassault Systèmes. Extension to the existing Dymola-Simulink interface.
dSPACE SCALEXIO	FMI_2.0				Available	dSPACE SCALEXIO is a Hardware-in-the-Loop (HIL) integration and simulation platform from dSPACE. Please also refer to the dSPACE FMI sites for more information about the FMI 1.0 and FMI 2.0 support.
dSPACE SYNECT	FMI_2.0				Available	dSPACE SYNECT is a data management tool from dSPACE that enables you to manage FMUs and Simulink models as well as their dependencies, versions and variants throughout the entire software development process. Please also refer to the dSPACE FMI sites for more information about the FMI support.
dSPACE VEOS	FMI_2.0				Available	dSPACE VEOS is a PC-based virtual integration and simulation platform from dSPACE. Please also refer to the dSPACE FMI sites for more information about the FMI 1.0 and FMI 2.0 support.
Dymola	FMI_2.0	Available 12	Available	Available 17	Available 21	Modelica environment from Dassault Systèmes.
ETAS - ASCMO	FMI_2.0			Available		Creation and export of statistical (meta) models using Design of Experiments (DoE) from ETAS.
FMI Blockset for Simulink	FMI_2.0				Available	Import of FMI Co-Simulation models into Simulink - provided by Claytex.
FMI Toolbox for MATLAB/Simulink	FMI_2.0	Planned	Available 23	Planned	Available 25	The FMI Toolbox for MATLAB/Simulink from Modelon enables FMU import and export for MATLAB/Simulink for both model exchange and co-simulation.
FMUSDK	FMI_2.0	Available 8		Available 8		FMU Software Development Kit from QTronic.
General Energy Systems (GES)	FMI_2.0	Planned	Planned	Planned	Planned	GES is an object oriented simulation tool, for dynamic and static (algebraic) systems, Based on a hybrid bondgraph model. The tool is mainly used for ship power designs. Provided by TNO.
GT-SUITE	FMI_2.0		Planned	Planned	Planned	Multi-Physics Simulation Platform for Powertrain and Vehicle Systems
MapleSim	FMI_2.0	Available 15	Planned	Available 15	Planned	Modelica-based modeling and simulation tool from Maplesoft
Silver	FMI_2.0		Available 24	Available 3	Available 30	Generation of virtual ECUs and virtual integration platform for Software in the Loop from QTronic.
SimulationX	FMI_2.0	Planned	Planned	Planned	Planned	Multi-domain simulation tool for design, analysis and virtual prototyping of complex systems by ITI.
xMOD	FMI_2.0		Available		Available	Heterogeneous model integration environment & virtual instrumentation and experimentation laboratory from IFPEN distributed by D2T.

FMI

- Some people think that FMI 1.0 is appropriate for **object oriented** modeling
- It supports traditional **block oriented** modeling!
- FMI 1.0 is intended for representing sub-systems with input/output causality
  - Binary code → Symbolic processing not possible
- Traditional Cosimulation problematic
  - No error control
  - No event handling

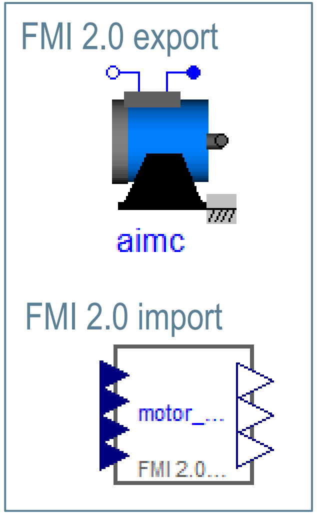
- FMI 2.0 Cosimulation
  - Error control possible
  - Interface Jacobain based Co-simulation
- FMI 2.1
  - HybridCosimulation with event handling

FMI 2.0 Released July 25, 2014

- **New features**
  - Many practical issues of FMI 1.0 fixed.
  - Parameters can be changed during simulation.
  - FMU state can be saved, restored, serialized.
  - Sparse structure of partial derivatives w.r.t states and inputs (large systems)
  - Directional derivatives w.r.t. states and inputs (large systems).
  - Algebraic loops over FMUs in all modes (initialization, event, continuous-time).
  - Co-Simulation can step back (e.g. many simulations from one time instant).
  - Cleaner mathematical model (e.g. super-dense time).
- **Tested with prototypes from 7 tool vendors**
  Dassault Systèmes, IFPEN, ITI, LMS-Imagine, Modelon, QTronic, OSMC
- **FMI 2.0 Plug-Fest, May 12-13, 2014**
  Evaluate and resolve compatibility issues between prototypes and FMI 2.0 RC2

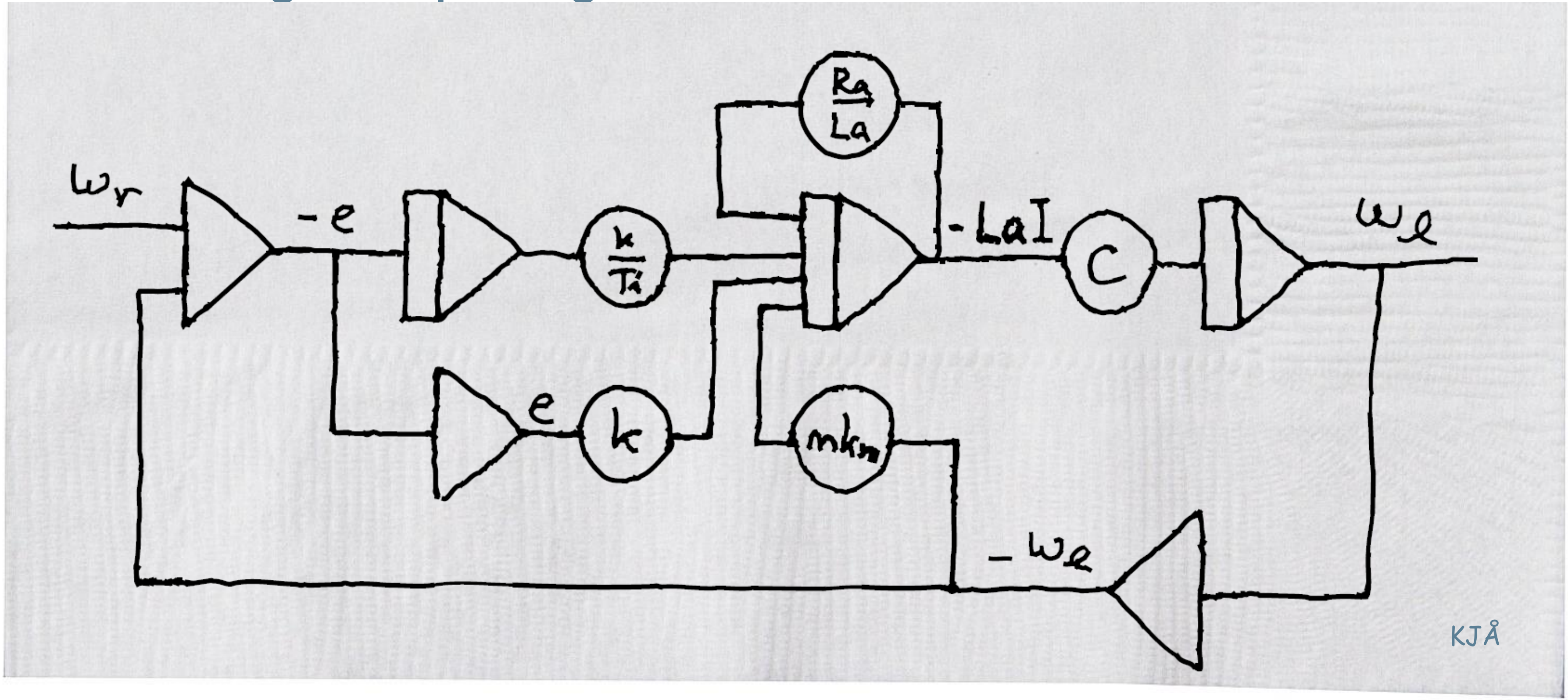
FMI 2.1 being developed

- **Export/Import FMUs without loss of information (separate compilation)**
  - Connectors
  - Hierarchical data structures
  - Clocks and discrete states
  - Residue equations in interface
  - Hybrid Co-simulation (cosimulation with event handling)
  - Local Index reduction
  - annotations (graphical view)
  - Partial derivatives with respect to parameters
  - Arrays with parameter dimensions
  - Changing number of states
- **Partial designs for missing features available (but incomplete, few prototypes)**

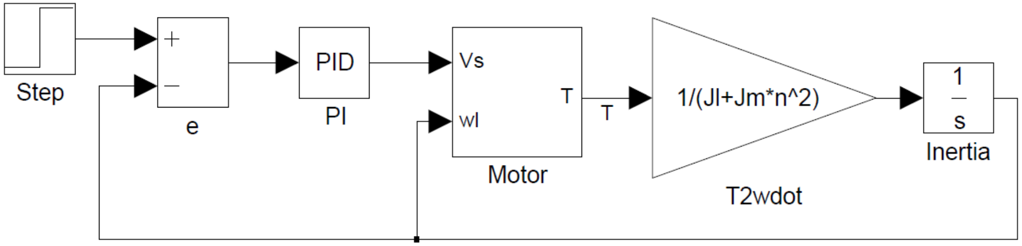


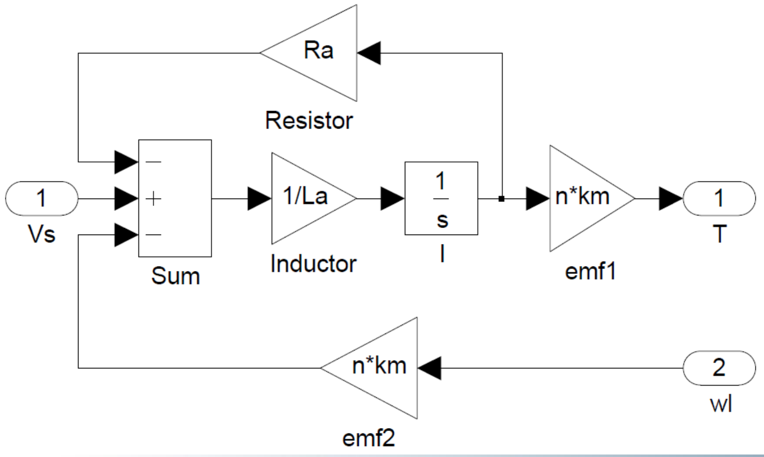
System Engineering Model Views and Experience

System Design on the Back-of-the-Napkin – Analog computing – 1940's-

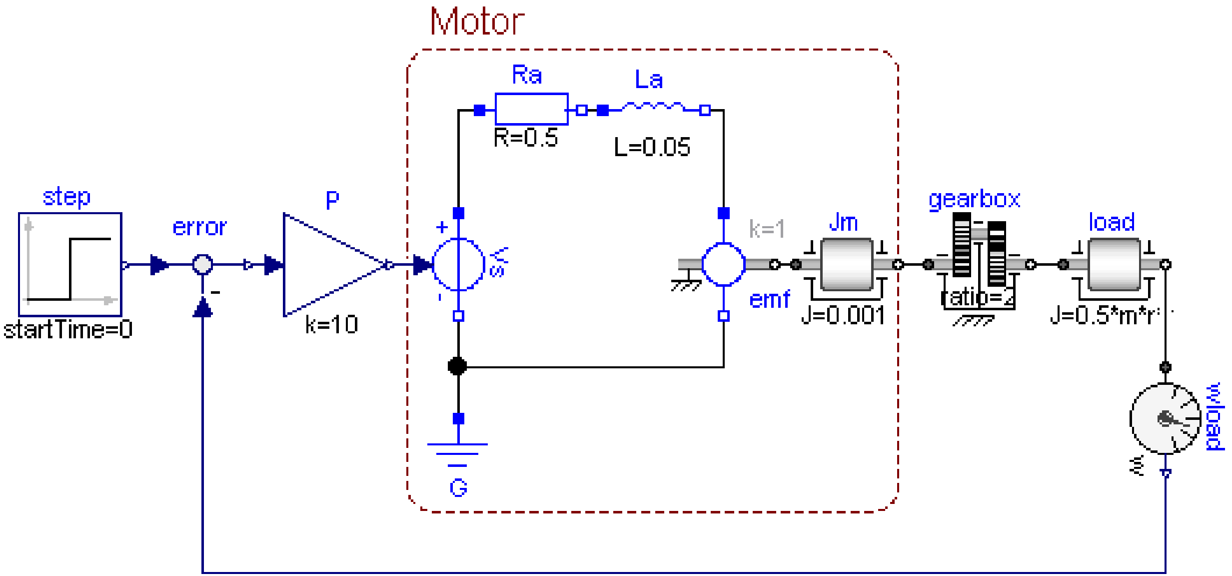


Next Step – 1980's - Block diagrams

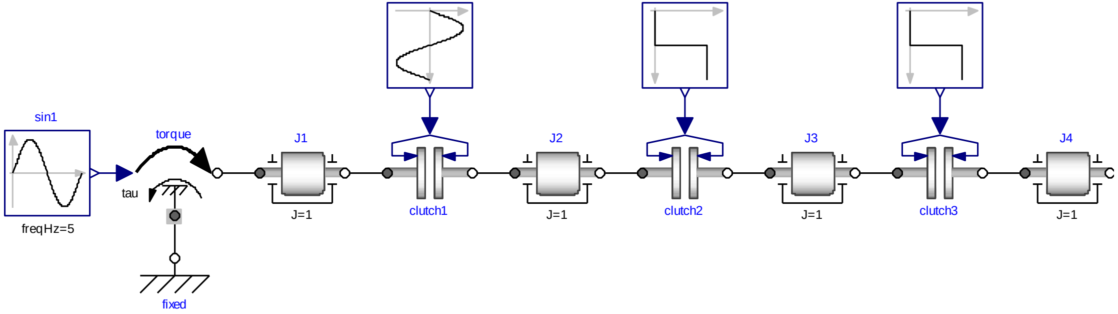




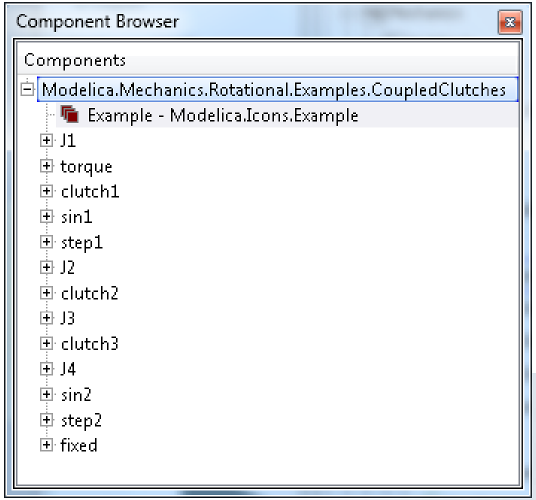
Next step – 1990's – Modelica acausal diagram

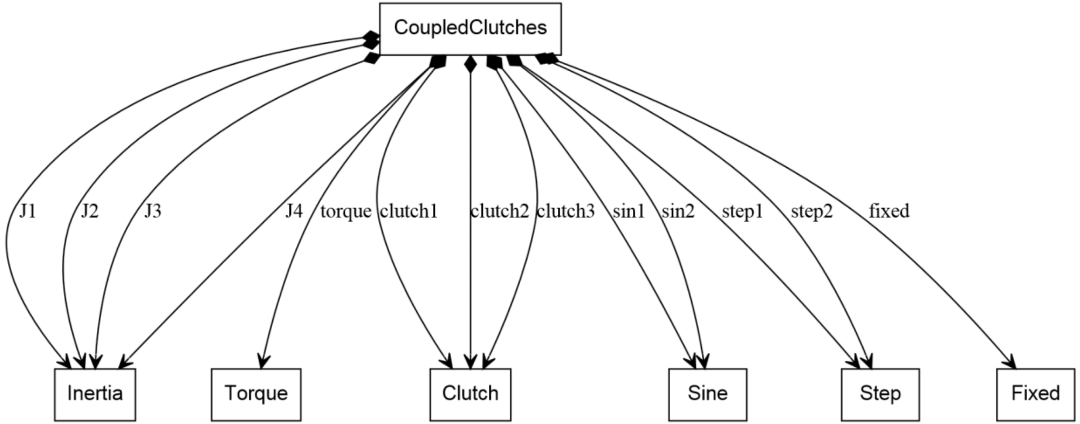


SysML – 2004 – Block Definition Diagram – BDD

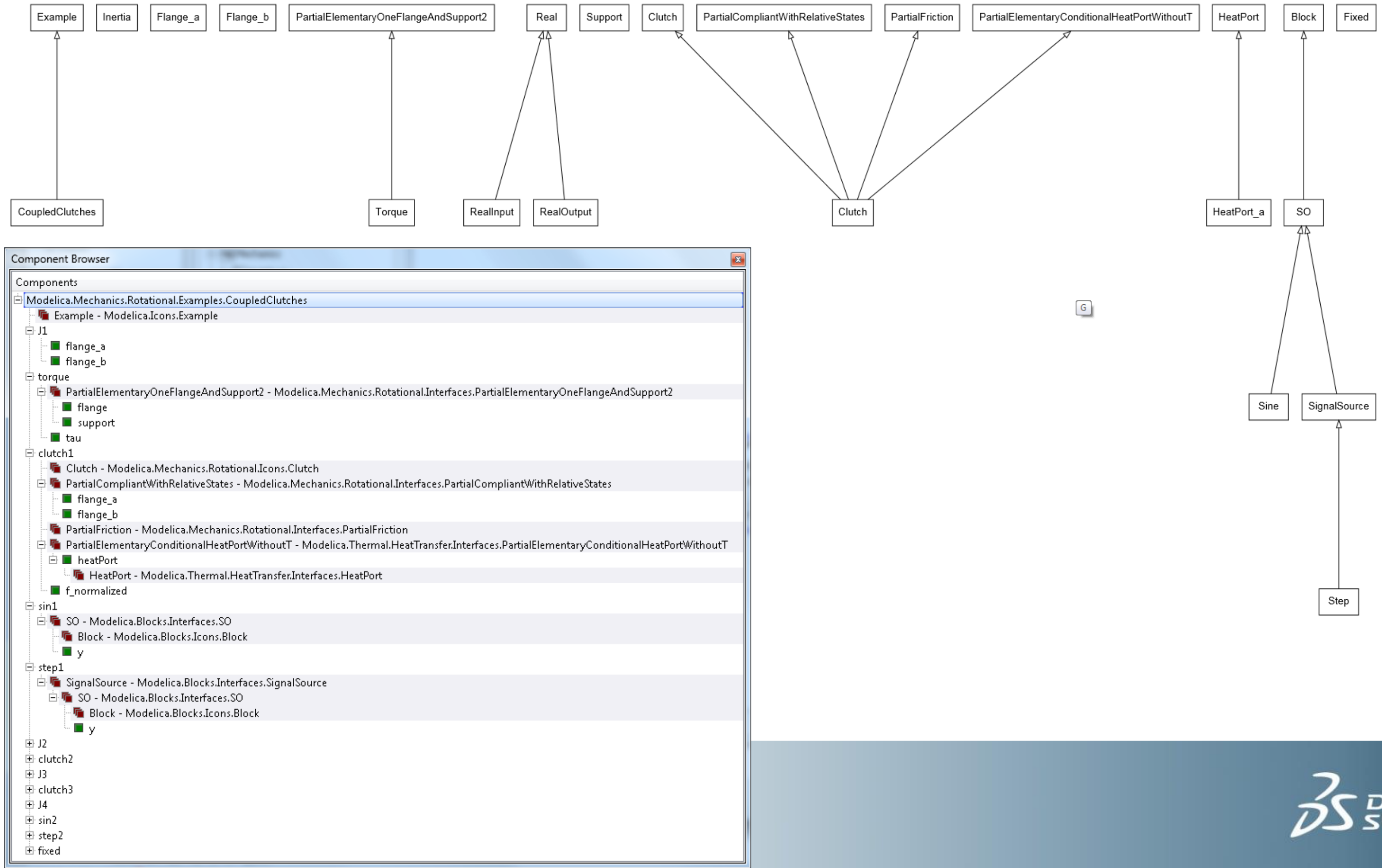


Modelica diagram corresponds to ibd (internal block diagram)

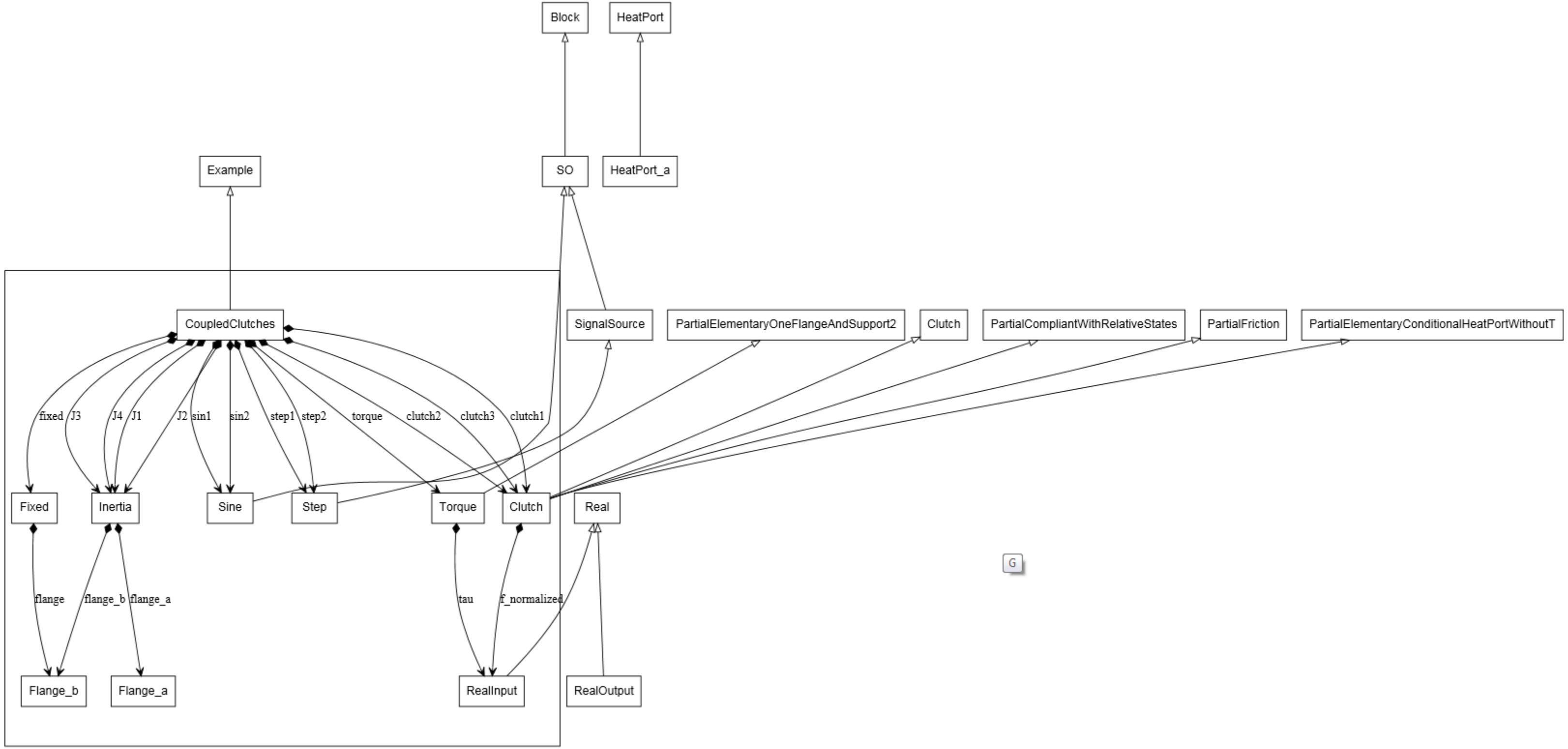




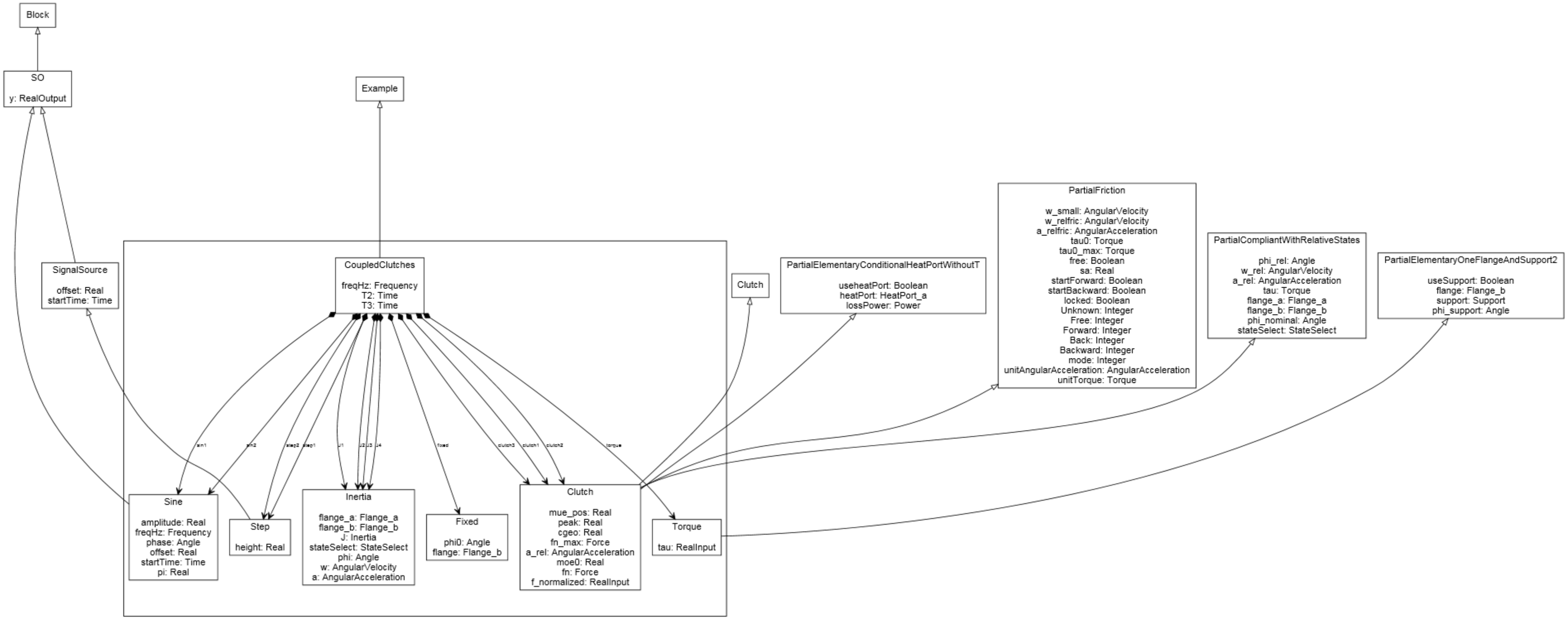
BDD - Inheritance



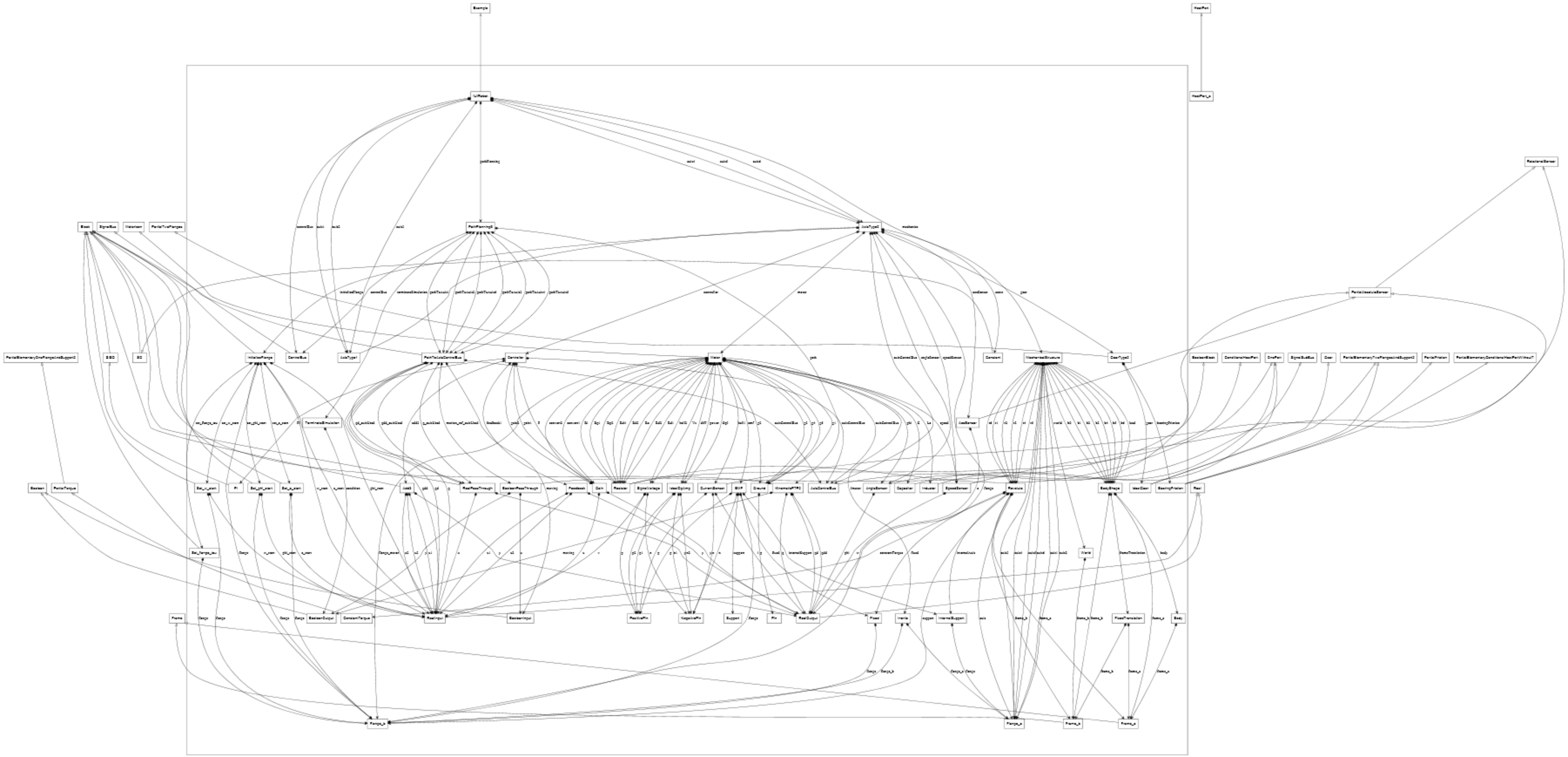
BDD – Components and Inheritance



BDD – Details

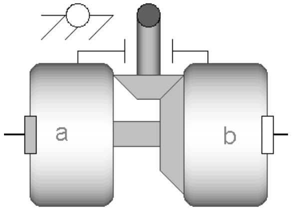

Block
SO
y: RealOutput
Example
SignalSource
offset: Real
startTime: Time
CoupledClutches
freqHz: Frequency
T2: Time
T3: Time
sin1
sin2
step2
step1
J1
J2
J3
J4
fixed
clutch3
clutch1
clutch2
torque
Sine
amplitude: Real
freqHz: Frequency
phase: Angle
offset: Real
startTime: Time
pi: Real
Step
height: Real
Inertia
flange_a: Flange_a
flange_b: Flange_b
J: Inertia
stateSelect: StateSelect
phi: Angle
w: AngularVelocity
a: AngularAcceleration
Fixed
phi0: Angle
flange: Flange_b
Clutch
mue_pos: Real
peak: Real
cgeo: Real
fn_max: Force
a_rel: AngularAcceleration
moe0: Real
fn: Force
f_normalized: RealInput
Torque
tau: RealInput
Clutch
PartialElementaryConditionalHeatPortWithoutT
useheatPort: Boolean
heatPort: HeatPort_a
lossPower: Power
PartialFriction
w_small: AngularVelocity
w_relfric: AngularVelocity
a_relfric: AngularAcceleration
tau0: Torque
tau0_max: Torque
free: Boolean
sa: Real
startForward: Boolean
startBackward: Boolean
locked: Boolean
Unknown: Integer
Free: Integer
Forward: Integer
Back: Integer
Backward: Integer
mode: Integer
unitAngularAcceleration: AngularAcceleration
unitTorque: Torque
PartialCompliantWithRelativeStates
phi_rel: Angle
w_rel: AngularVelocity
a_rel: AngularAcceleration
tau: Torque
flange_a: Flange_a
flange_b: Flange_b
phi_nominal: Angle
stateSelect: StateSelect
PartialElementaryOneFlangeAndSupport2
useSupport: Boolean
flange: Flange_b
support: Support
phi_support: Angle

BDD - Robot

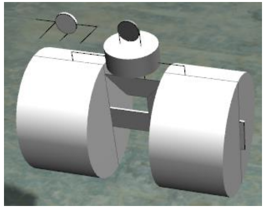


Next step - 2015

- More modern experience to make modeling, simulation and systems engineering popular
- Young students are used to 3D games
- Proposal to enhance Modelica with 3D schematics



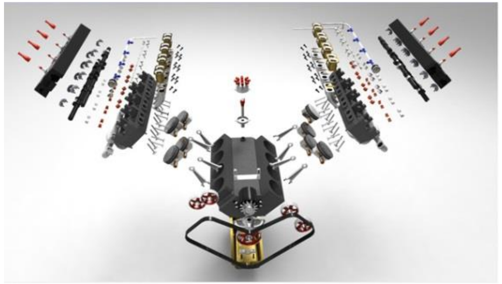
Modelica Icon from 90’s



Reinterpretation in 3D

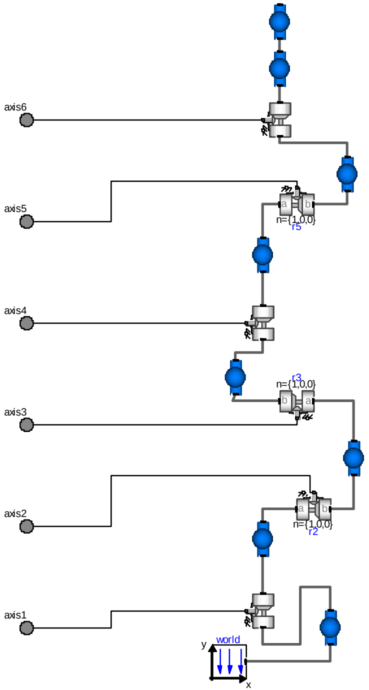
Unification of views – 3D Schematics

- Assembly of 3D CAD parts
  - Joints not seen
- CAD Exploded view

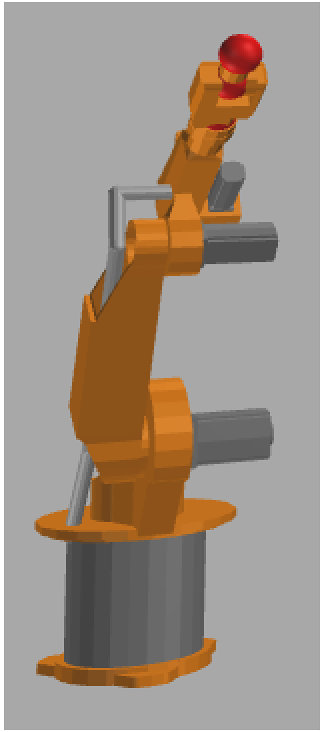


- Idea
  - Explode view only for joints
  - Along axis of motion
  - Show joints
  - Allow defining additional data (friction, etc) by selecting joint
  - Allow connecting other components such as electrical or hydraulic motors to joints

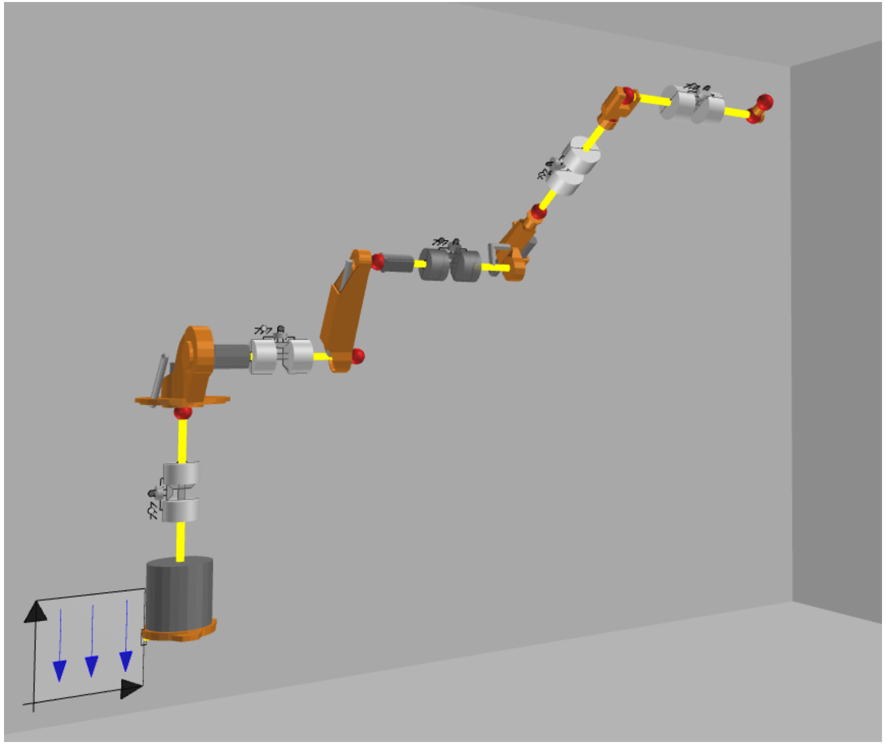
3D Schematics – Assembly/Exploded view – 2015



MSL - Today



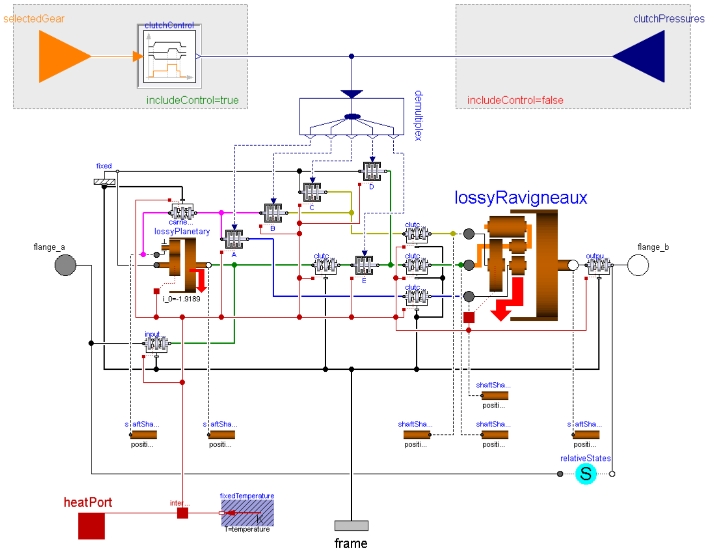
Assembly



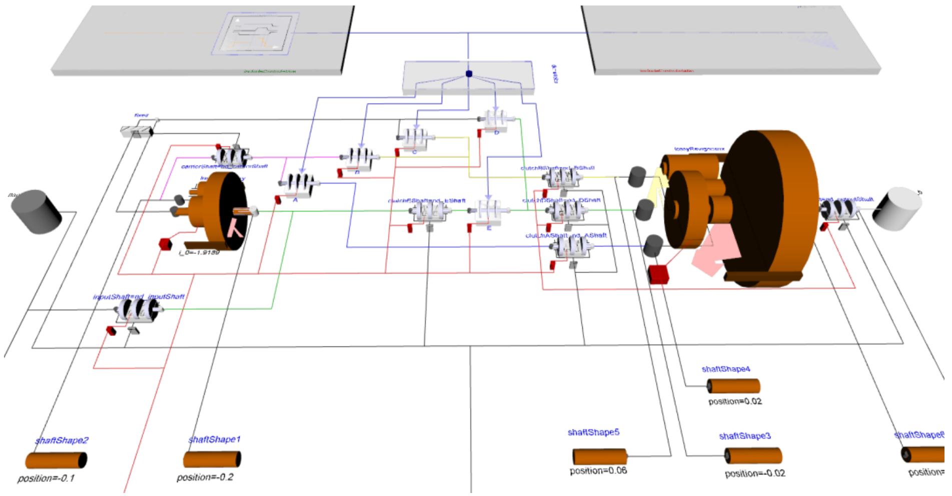
Exploded

3D Schematics - Gearbox

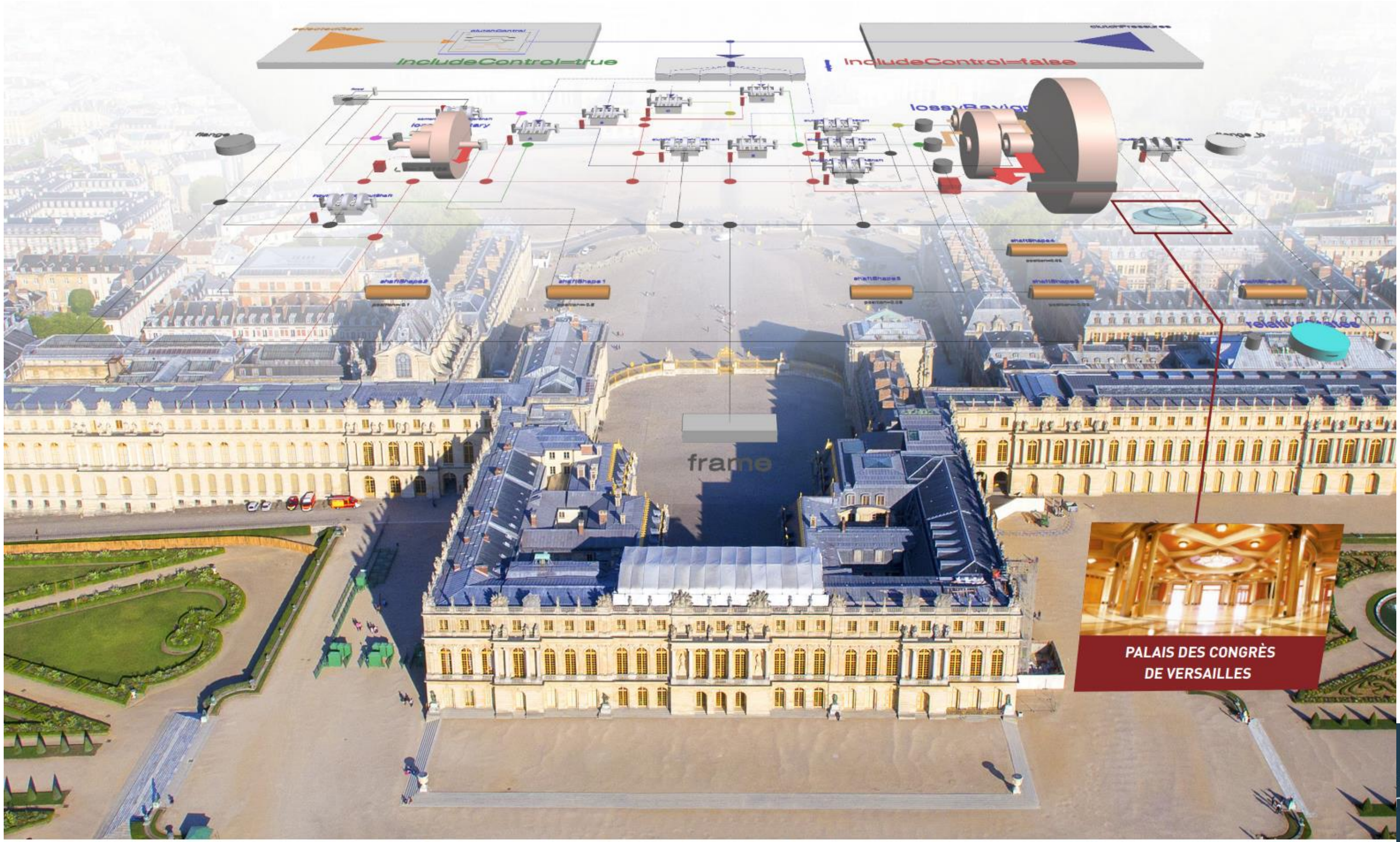
Modelica diagram today



Modelica 3D schematics



Modelica Conference 2015 - Poster



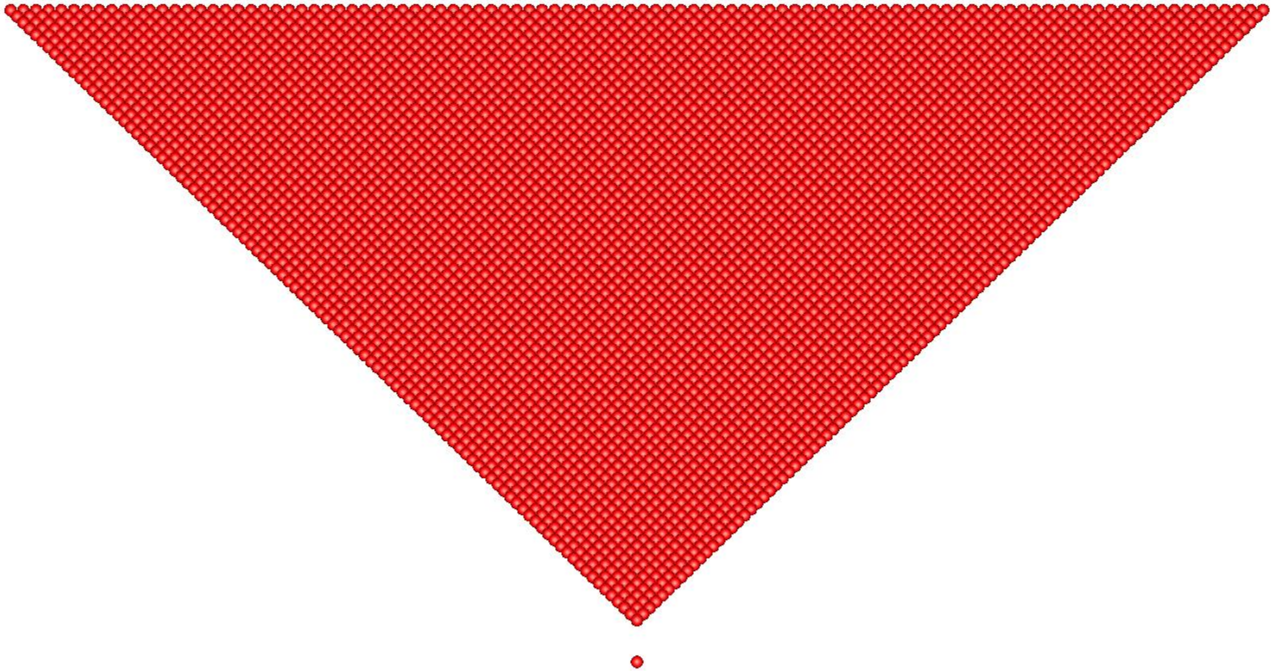
Modelica Conference 2015

- Palais de Congrès - Versailles, Paris
- September 21-23, 2015
- **Paper Deadline May 20**
- Organized by
  - Modelica Association
  - Dassault Systèmes and Linköping University
- Conference Chair: Hilding Elmqvist
- Program Chair: Peter Fritzson
- Welcome!

Conclusion

- Modelica is intuitive, convenient and secure
  - Graphical icons according to physical appearance
  - Connecting components according to physics
  - Component models uses physical equations (mass-, energy-balances, etc.)
- 3DEXPERIENCE gives coupling between Modelica and 3D
- FMI for separate compilation and tool coupling
  - FMI 2.x convenient and secure
- A modern 3D systems engineering experience needed

A model – Discrete Element Method



A Modelica model

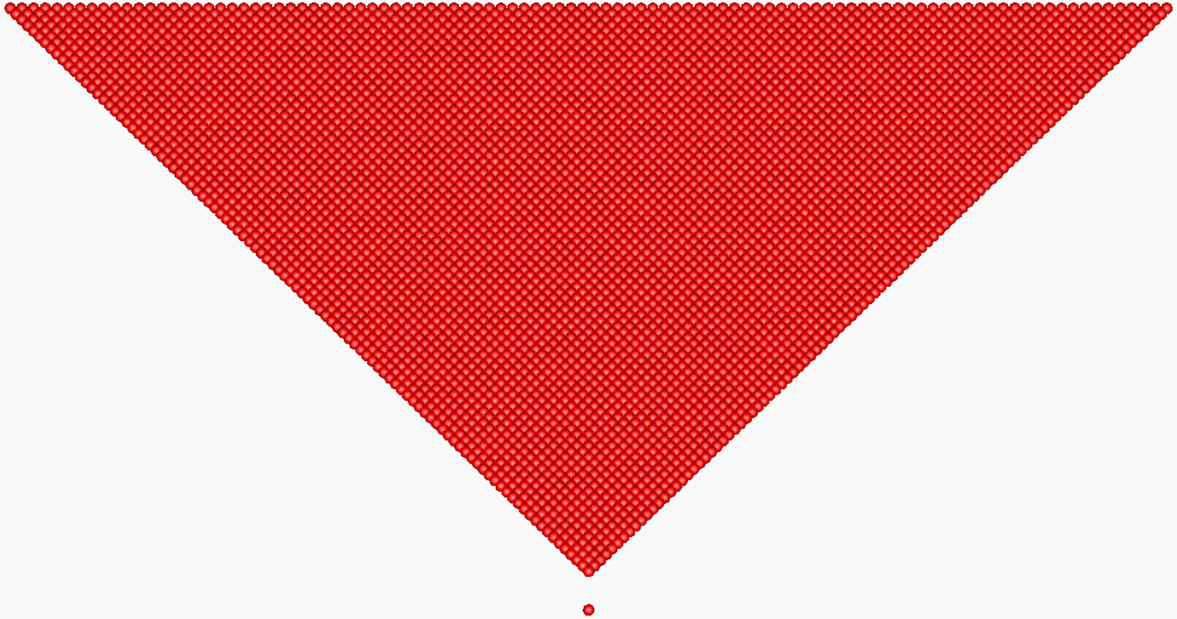
```
model Billiard
  parameter Integer layers=100;

  parameter Integer n=div(layers*(layers+1),2);
  inner CollidingWorld collidingWorld;
  Sphere sphere[n](x0={{layer(i)*sqrt(3)/2, column(i)-(layer(i)-1)*0.5, 0} for i in 1:n}, each radius=0.5) ;
  Sphere sphere1(x0={-5,0,0}, v0={1,0,0}, radius=0.5);
end Billiard;
```

5 050 balls

12.5 Million possible collision pairs

30 300 state variables



IF WE™