Modelica Library & Dynamic simulators

Aplicaciones en Informática Avanzada S.L. (AIA)

Réseau de transport d'électricité (RTE).

iPST session, June 01st 2017

Outline

- Introduction
- iPSL library
- Power Systems on Modelica (PSM) tool
- Demo
- Next steps & challenges

Introduction

- This project stems from the **needs for dynamic power system simulations** towards a single electricity market:
  - Open source and “white-box” tools.
  - Pan-European collaboration.
  - Adaptability to a fast-changing environment:
    - New DC current and renewable energy sources.
    - Growing diversity and complexity of automata.
- **Modelica** seems to be a promising modelling language for power system simulations:
  - Non proprietary.
  - Generic and powerful.
  - Equation based modelling language to conveniently model systems.
- **RTE-AIA project aims to open a path towards solving these needs** (open source tool, user friendly, flexible, easy generation of Modelica systems…)

Introduction

- Modelica is a standard generic modelling language:
  - Open language normalized by an association: any tool may support it (unlike Eurostag, MATLAB/Simulink formats).
  - Widely used, mostly in the automotive industry (Dassault, Airbus, ABB, Siemens).
  - Allowed in CIM CGMES user-defined models.

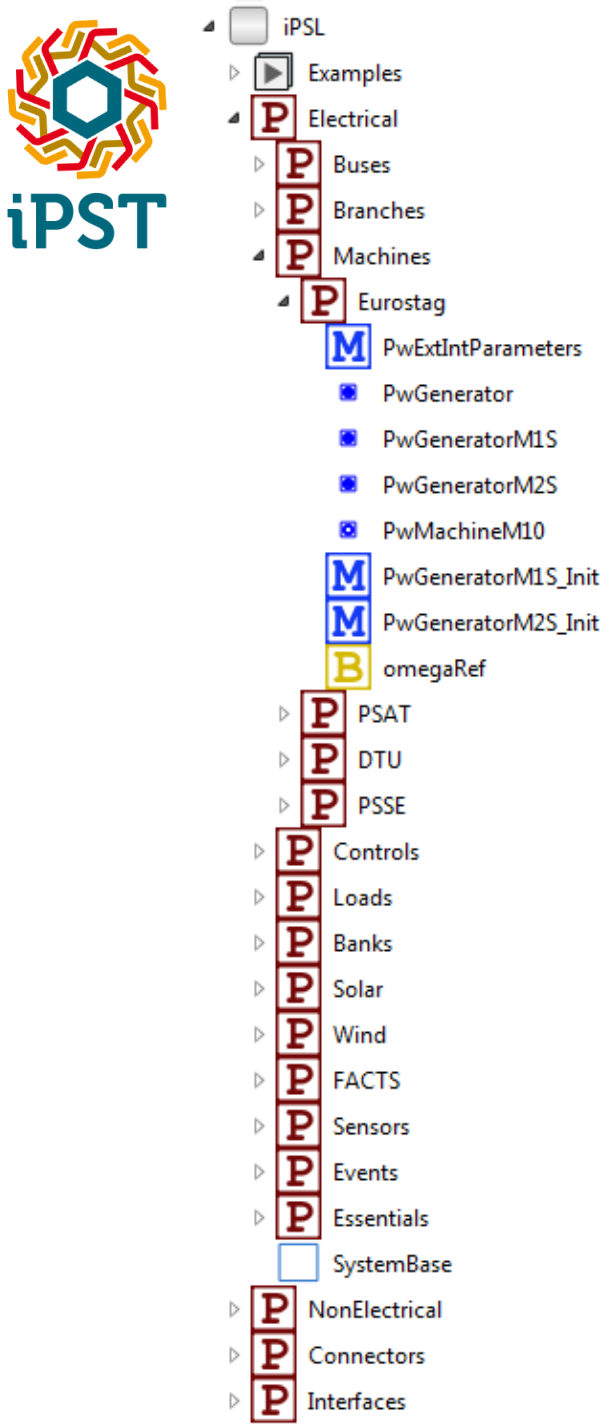
- Designed for easy modelling of complex cyber-physical systems: high level code, close to actual physical setup:
  - "Textbook" equations, "physical" connections.
  - Easier to write, reuse, maintain.

- But no Modelica tool tailored for power-system studies:
  - CIM data compatibility.
  - Load flow.
  - Dynamic data base.

- However, Modelica tools are not optimized for power system setups.

iPSL library

- A Modelica library called **iTesla Power System Library** (iPSL) was developed by different partners during the **iTesla** project.
- The library contains a set of power system component models for phasor time domain simulations.
- It also includes mathematical models for control systems.
- iPSL models were validated against domain-specific simulations tools (PSS/E and Eurostag) during the project.
- iPSL is one of the open-source projects on **GitHub iTesla repository**: https://github.com/itesla/ipsl
- iPSL was recently added as a power system Modelica library available with **OpenModelica** environment on February 2017.
- It is currently being maintained by AIA with full support from RTE.
- iPSL currently uses many models from Modelica standard library.



iPSL library

iPSL package contains 4 main packages:

- Electrical package
- Non Electrical package
- Connectors
- Interfaces

```
within iPSL.Electrical.Loads.Eurostag;
model PwLoadVoltageDependence
  "Load with voltage dependence.Developed by AIA. 2014/03/10"
  extends iPSL.Electrical.Essentials.pfComponent;
  iPSL.Connectors.PwPin p(vr(start=Vo_real),  vi(start=Vo_img), ir(start=ir0), ii(start =
ii0)) annotation(Placement(transformation(extent = {{-80, 0}, {-60, 20}}),
iconTransformation(extent = {{-80, 0}, {-60, 20}})));
  parameter Real Vo_real = V_0 * cos(angle_0 * Modelica.Constants.pi / 180) "Initial voltage
at node in p.u. (Real part)";
  parameter Real Vo_img = V_0 * sin(angle_0 * Modelica.Constants.pi / 180) "Initial voltage at
node in p.u. (Imaginary part)";
  parameter Real vo = sqrt(Vo_real ^ 2 + Vo_img ^ 2);
  parameter Real alpha = 0;
  parameter Real beta = 0;
  Real P(start = P_0/S_b);
  Real Q(start = Q_0/S_b);
  iPSL.Interfaces.AddedConnector P_1;
  iPSL.Interfaces.AddedConnector Q_1;
  parameter Real ir0=(P_0*Vo_real + Q_0*Vo_img)/((Vo_real^2 + Vo_img^2)*S_b) "Initialitation";
  parameter Real ii0=(P_0*Vo_img - Q_0*Vo_real)/((Vo_real^2 + Vo_img^2)*S_b)
"Initialitation";
protected
  Real a(start = 1) "auxiliary variable. Voltage division";
  Complex V(re = p.vr, im = p.vi);
  Complex I(re = p.ir, im = p.ii);
  Complex S;
equation
   P_1.y = P_0;
   Q_1.y = Q_0;
   a =  ComplexMath.'abs' (V) / vo;
   S.re = ((P_0 + P_1.deltaY)/ S_b) * (a^alpha);
   S.im = ((Q_0 + Q_1.deltaY)/ S_b) * (a^beta);
   S =  V*Modelica.ComplexMath.conj(I);
   P = p.vr * p.ir + p.vi * p.ii;
   Q = (-p.vr * p.ii) + p.vi * p.ir;

  annotation ( ...);
end PwLoadVoltageDependence;
```

PSM tool

- The main objective of RTE and AIA collaboration project is the development of an **open source tool** that provides an **easy way to prepare and perform dynamic simulations using Modelica.**
- This goal stands on a solid background:
  - Developments and experience acquired during the FP7 EU project **iTesla**, on the automatic transformation of power system networks from different proprietary format into Modelica language.
  - The iTesla project served as a *proof of concept* for the usage of Modelica as a standard and common language for power systems modelling and simulation.
  - Several European power systems ranging from a dozen to hundreds of buses were successfully converted from PSS/E or Eurostag format into Modelica.

PSM tool

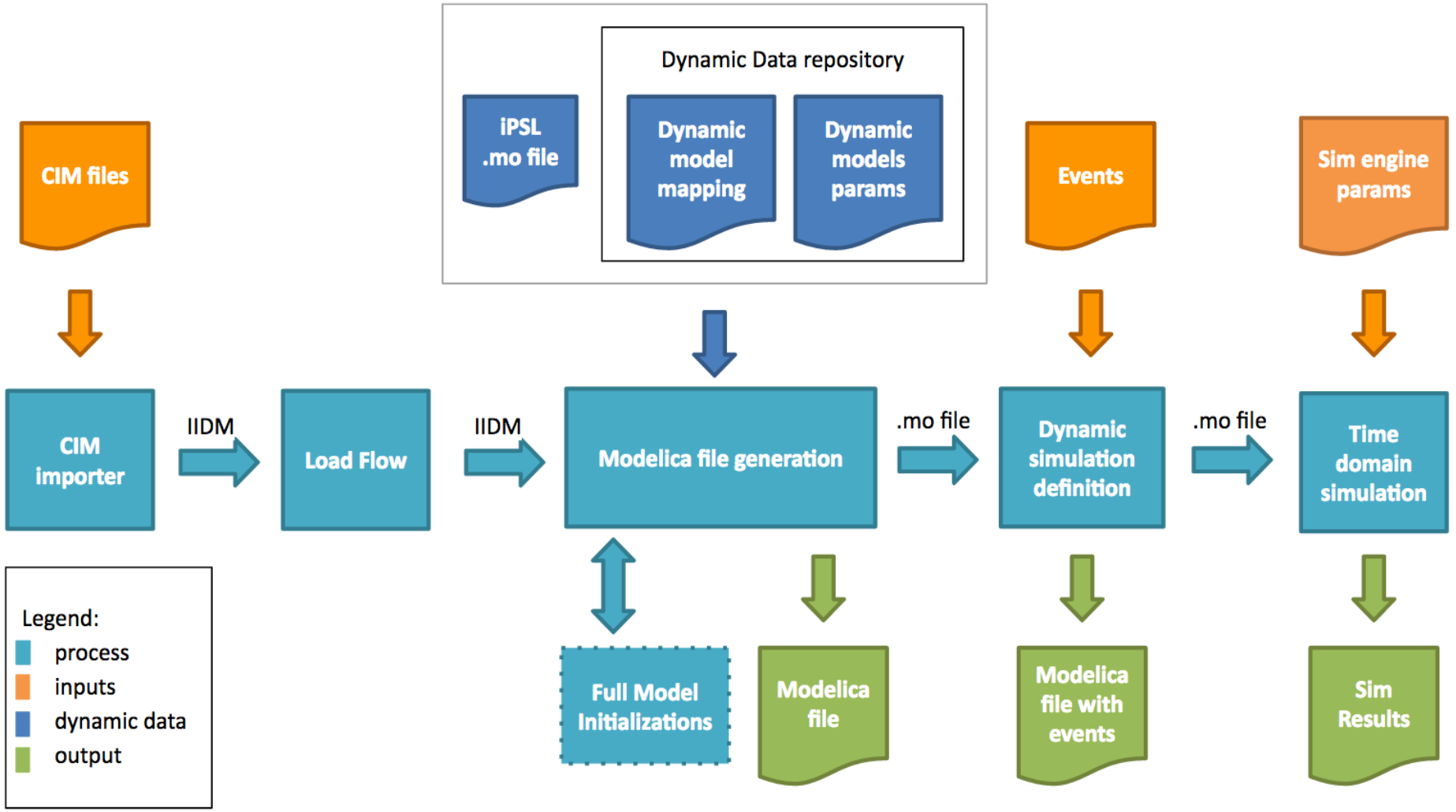
- The tool is currently named **PSM**: Power Systems on Modelica.
- The tool allows the users to **transform** power system networks **from CIM** standard format **to Modelica** (*.mo files).
- PSM runs a Load Flow computation and uses a Dynamic Data Repository to retrieve dynamic models.
  - Two alternatives are available for loadflow computation: **Hades2** (RTE) and **HELM-Flow** (AIA) engines.
- PSM is being designed to be **fully compatible with iPSL** library but the user is not limited to the library models.
- Dynamic simulations can be run inside the tool on the generated Modelica files, using **OpenModelica** (open source) or **Dymola** (commercial) engine.

PSM tool

- It will be released as **open source**, for Linux and Windows OS.
- It has been designed to ensure **modularity**, allowing to either run processes individually or as a full workflow.
- A **desktop GUI** has been developed to allow the user to select the processes to run, on which data sets, and display progress and logs.
- **Command line tools** are also available to run the tool.
- A recent publication about PSM has been published in the Proceedings of the **12th International Modelica Conference 2017**:
  - *A Modelica-based Tool for Power System Dynamic Simulations*:

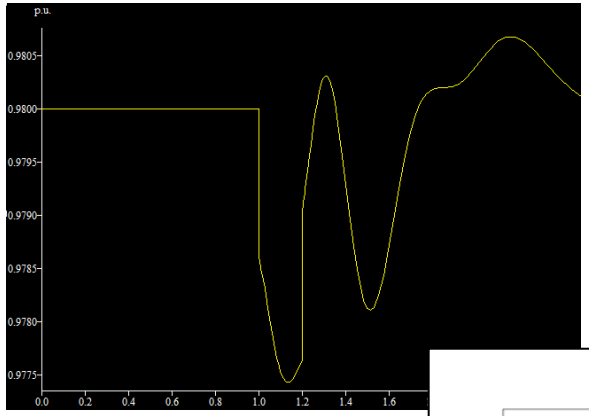
https://modelica.org/events/modelica2017/proceedings/html/authors/author_22.html

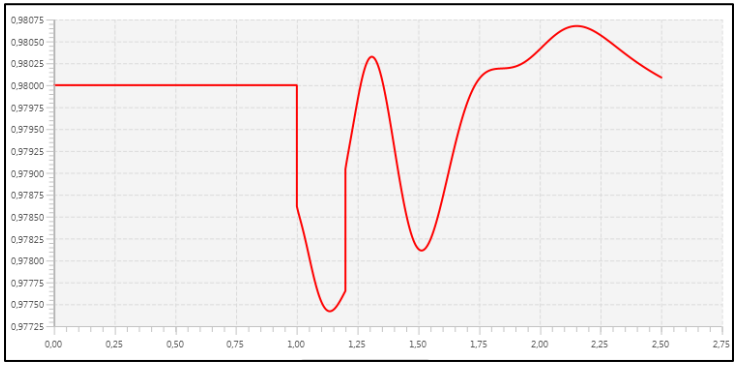
PSM tool

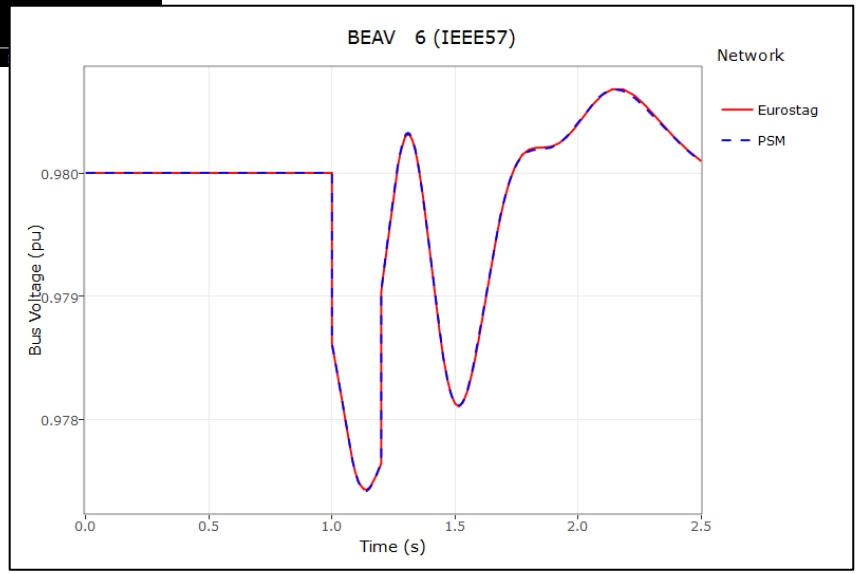

Dynamic Data repository
iPSL .mo file
Dynamic model mapping
Dynamic models params
CIM files
Events
Sim engine params
CIM importer
IIDM
Load Flow
IIDM
Modelica file generation
.mo file
Dynamic simulation definition
.mo file
Time domain simulation
Full Model Initializations
Modelica file
Modelica file with events
Sim Results
Legend:
process
inputs
dynamic data
output

PSM tool

- Dynamic simulations performed on generated Modelica file for IEEE57 case accurately reproduce Eurostag results.







Demo

Next steps & challenges

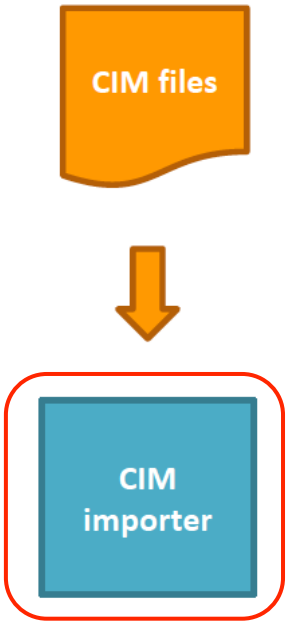
- PSM will be delivered to the public on **June 2017**.
- It will be distributed with public networks included.
- Users will have the possibility to use PSM with available cases or new cases using dynamic user-defined models.
- RTE and AIA expect to have a lot of feedback from users.
- iPSL library will suffer a large improvement in the near future that will have a direct impact in the simulation speed. PSM will be updated when ready.
- **Scalability**: PSM allows to convert to Modelica networks of any size but…
  - Modelica simulation engines performance is highly decreased with networks above 100 nodes (very far from realistic models).
  - Future works should go on this direction.
- Future improvements should also be directed by users feedback.

Thank you!

Introduction

```
1  model DYNModelGeneratorPV_flat "Generator with pre-set active power and voltage outputs" // 14 unknowns, 12 equations
2    import Modelica.ComplexMath;
3    import Modelica.Blocks.Interfaces;
4    import Connectors;
5  public
6      parameter SIunits.ActivePower P0 "initial active power supplied";
7      parameter SIunits.ReactivePower Q0 "initial reactive power supplied";
8      parameter Real lambda( unit = "kV/MVAr")  "voltage sensitivity of reactive power regulation";
9      parameter SIunits.ReactivePower QMin "minimum reactive power";
10     parameter SIunits.ReactivePower QMax "maximum reactive power";
11     parameter SIunits.ActivePower PMin "minimum active power";
12     parameter SIunits.ActivePower PMax "minimum reactive power";
13     Connectors.ACPower out "voltage and current output"
14+    annotation (Placement(transformation(extent={{100,-10},{120,10}}), ...);
16     Interfaces.RealInput UControl "control voltage. this variable is set in order to allower the user to change this set-point
   during the simulation." annotation (Placement(transformation(extent={{-140,-20},{-100,20}})));
17 protected
18     Real PRaw "active power output without taking maximum and minimum power into account";
19     Real P( start = P0) "active power";
20     Real QControl "reactive power target";
21     Real Q( start = Q0) "reactive power";
22     Complex S "apparent power";
23     Real VNw;
24     parameter Real QDeadBand = 1e-4 "reactive power dead-band";
25     parameter Real VDeadBand = 1e-4 "voltage dead-band";
26 equation
27     PRaw = P0;
28     S = Complex(P, Q); //managing complex numbers
29     VNw = ComplexMath.'abs'(out.V);
30     UControl - VNw = lambda * QControl;
31     P = if PRaw >= PMax then PMax
32       elseif PRaw <= PMin then PMin
33       else PRaw;
34     Q = if (QControl <= QMin) then QMin
35       elseif (QControl >= QMax) then QMax
36       else QControl;
37     assert( Q <= QMax, "Q is too big");
38     S = - out.V * ComplexMath.conj( out.i); // the current is positive when entering the device
39+   annotation (uses(Modelica(version="3.2.2")), ...);
53 end DYNModelGeneratorPV_flat;
```

PSM tool: Workflow

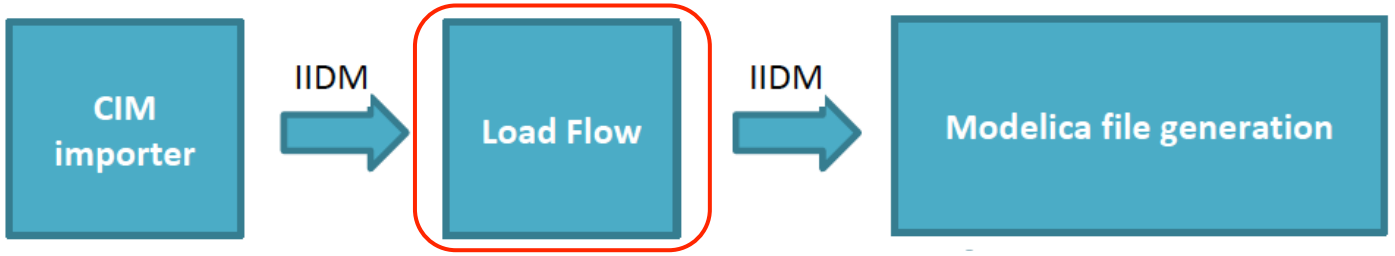


1. CIM importer

Imports the network system model file from CIM format (CIM CGMES very soon) and convert it into IIDM (iTesla Internal Data Model).

2. Load Flow computation

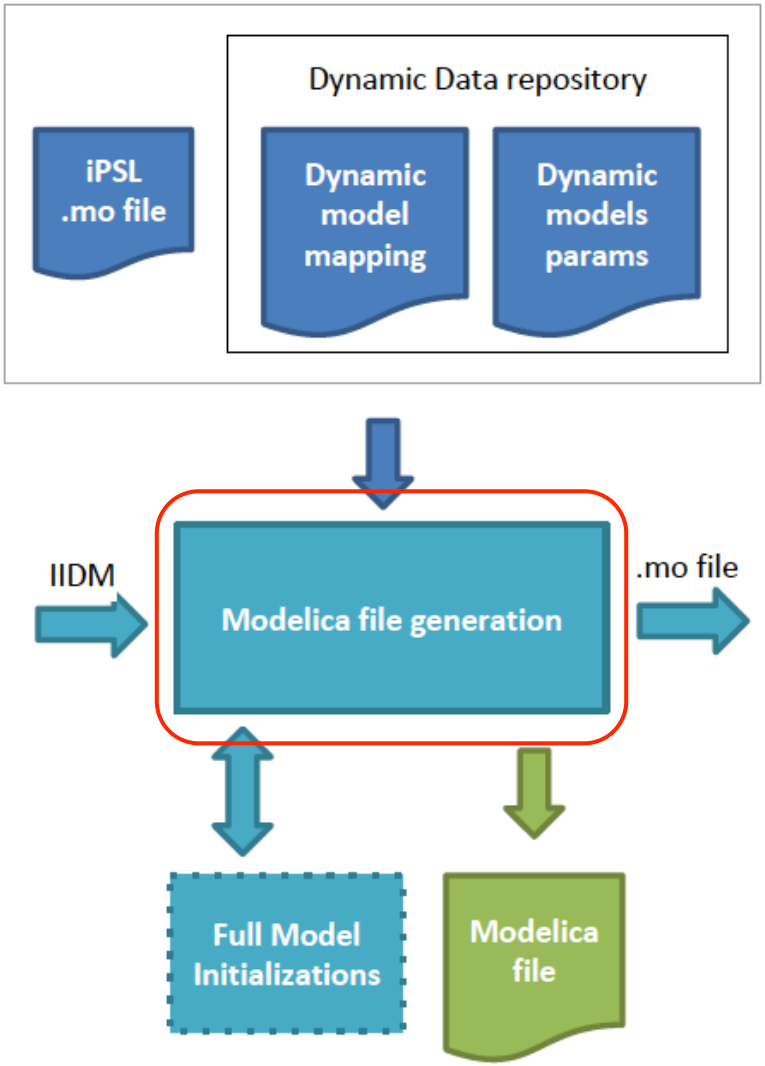
- Runs the Load Flow over the IIDM network obtained in the previous step and results are re-injected into this IIDM.
- Two alternatives:
  - Hades2.
  - HELM™-Flow.



PSM tool: Workflow

3. Modelica file generation

- Generates the Modelica file (base case) by following these steps:
  1. Selection of a Dynamic Data Repository (DDR).
  2. Mapping of dynamic models with static objects based on various criteria
  3. Build multiple Modelica documents for full model initializations.



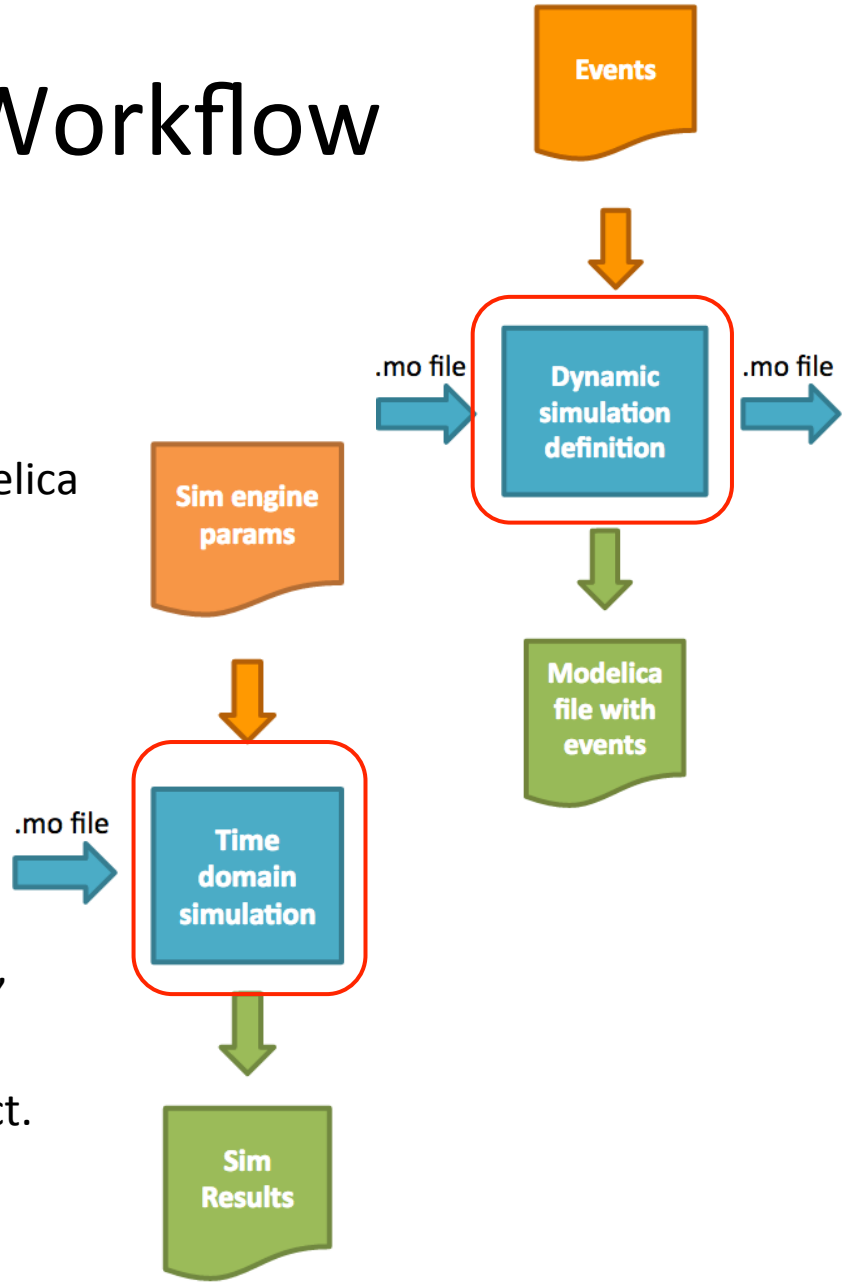
PSM tool: Workflow

4. Dynamic simulation definition

- Definition of the simulation scenario.
- Events can be added to the obtained Modelica system model.

5. Time-domain simulation

- Run the time-domain simulation.
- Two alternatives:
  - OpenModelica, that is an open source, community driven initiative.
  - Dymola, which is a commercial product.



PSM tool: Workflow

Dynamic Data Repository (DDR)

- A DDR is used for defining which dynamic models (from iPSL library or user-defined) should be used to map network equipments, and which parameters must be used to instantiate those dynamic models.
- Implementation based on **XML files** and use of **templates** (same model definition applied to a group of elements).
- Structure borrowed from RTE dynamic mapping (close to Dynamo).