Modeling with Modelica

OpenIPSL

A Modelica Library for Power System Simulation

Prof. Luigi Vanfretti

luigiv@kth.se, https://www.kth.se/profile/luigiv/

This work was supported in part by:

openCPS

ITEA3 EUREKA innovation across borders

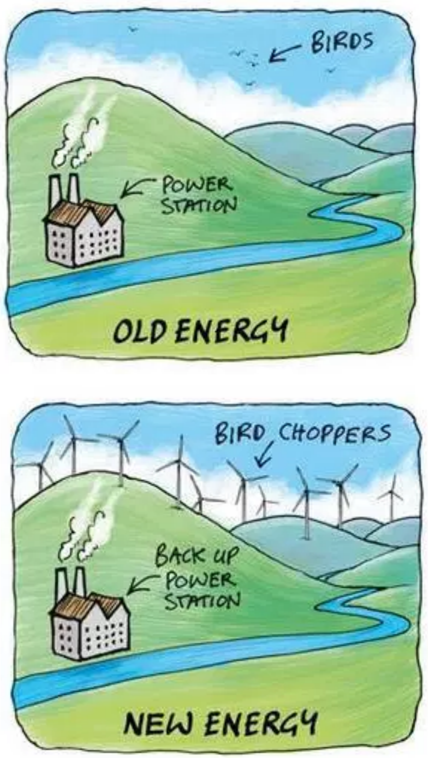
STandUP for ENERGY

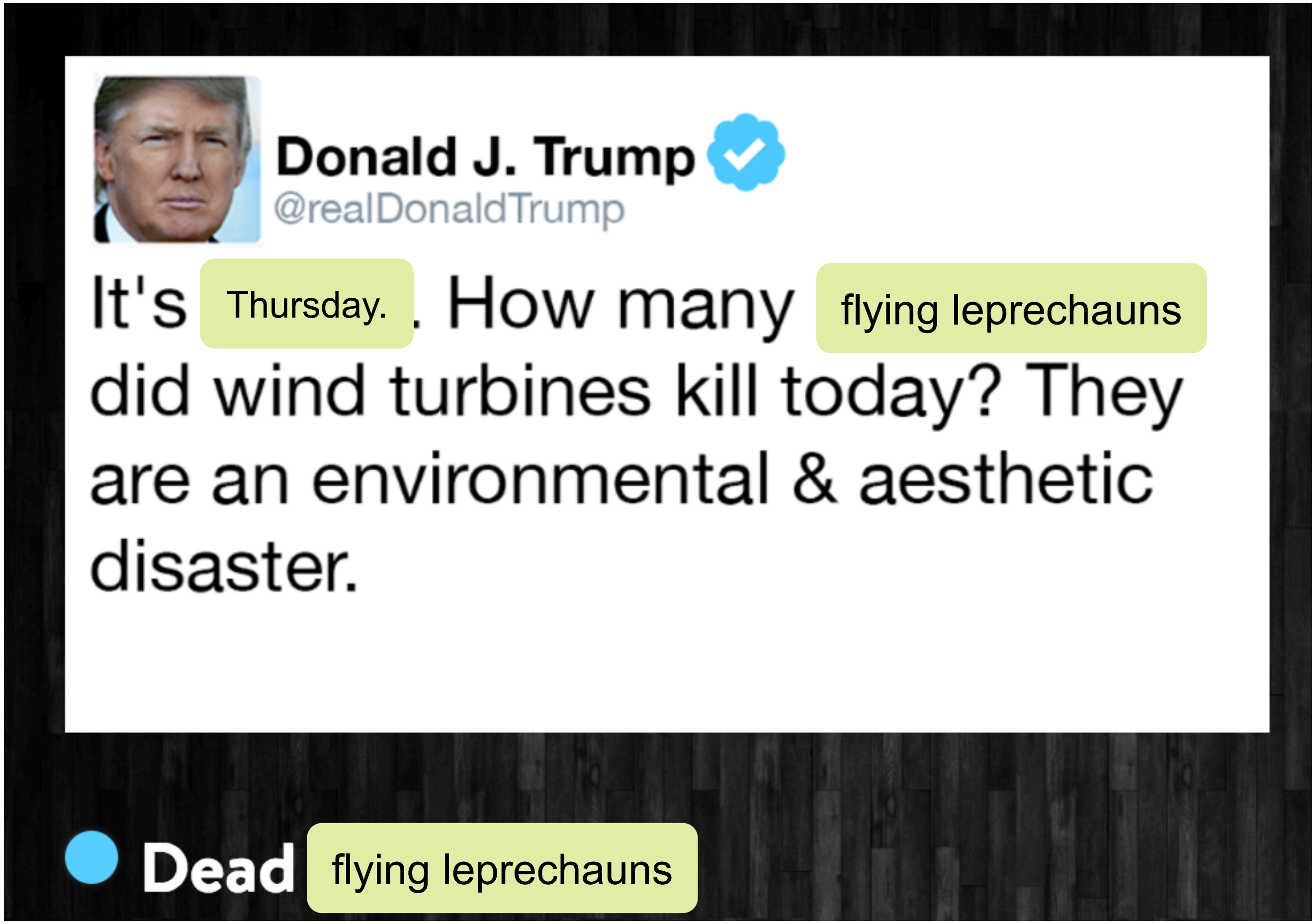
2ND WORKSHOP ON DYNAMIC SYSTEM MODELLING

March 23rd, 2017
Dublin, Ireland

I heard wind power is big deal in Ireland…

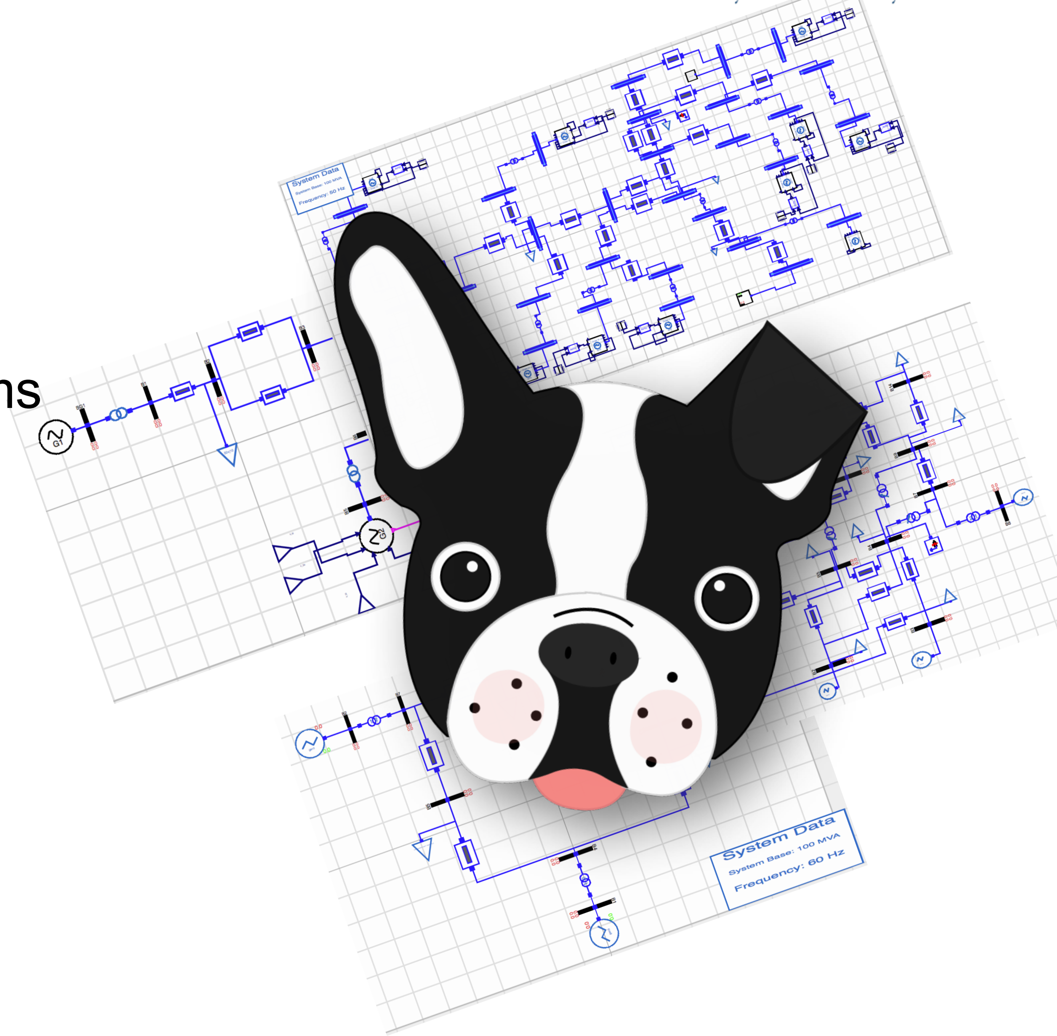
The news have traveled far indeed!





Outline

- Generalities and the role of models and simulation
- Modelica and power systems
- OpenIPSL
- Project documentation
- On-going developments



The Underlying Question:

Why do we develop models and perform simulations?

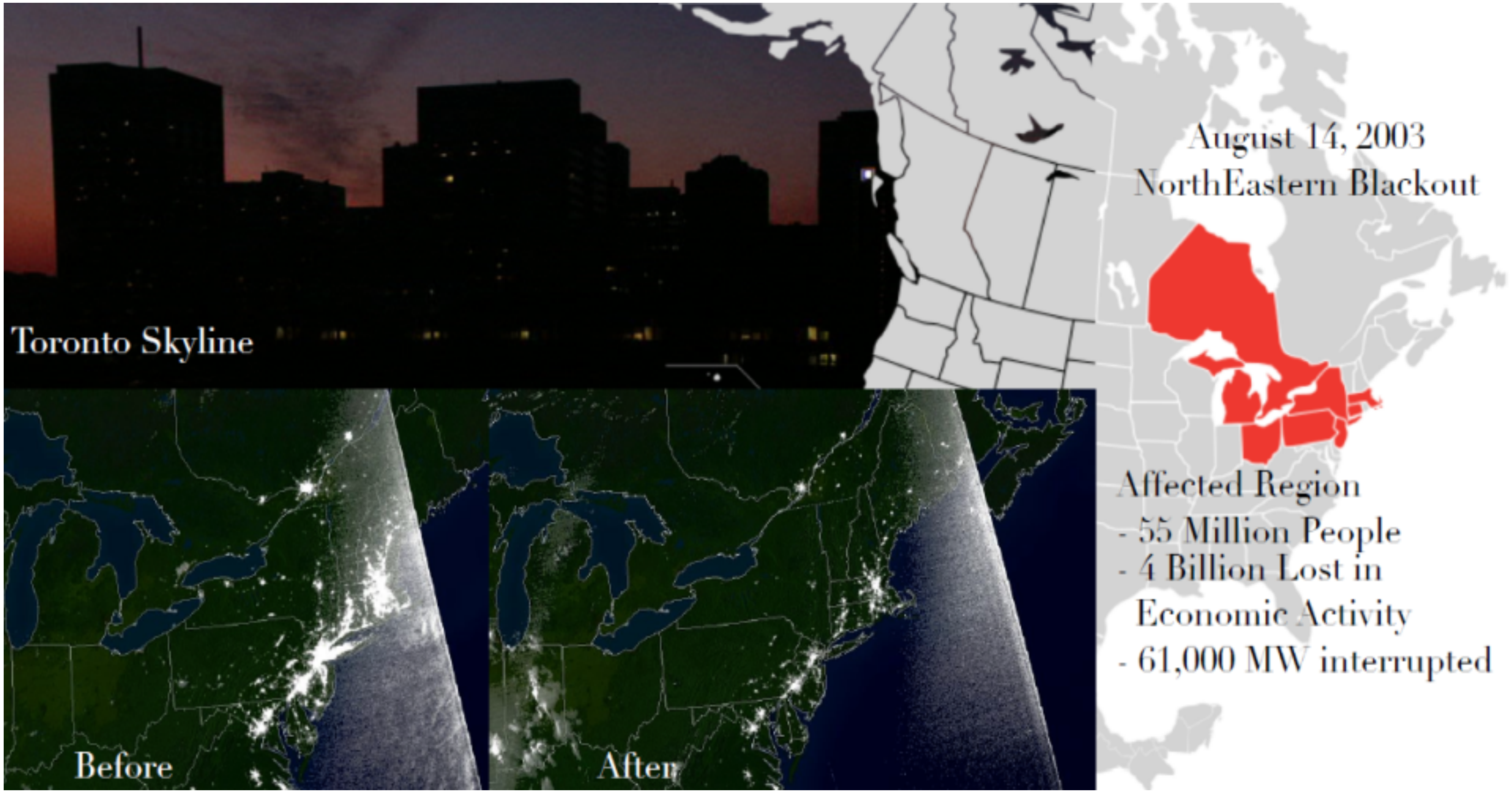
To reduce the lifetime cost of a system

- *In requirements:* trade-off studies
- *In test and design:* fewer proto-types
- *In training:* avoid accidents
- ***In operation:* anticipate problems**

- The prospective pilot sat in the top section of this device and was required to line up a reference bar with the horizon. 1910.
- More than half the pilots who died in WW1 were killed in training.

A **Failure** to Anticipate → Huge Costs!

There are many examples of failures to anticipate problems in power system operation



Others: WECC 1996 Break-up, European Blackout (4-Nov.-2006), London (28-Aug-2003), Italy (28-Sep.-2003), Denmark/Sweden (23-Sep.-2003)

Failure!: Existing modeling and simulation (and associated) tools were unable to predict these events.

The Multiple Roles of Modeling and Simulation
in building Complex Cyber-Physical ”Systems-of-Systems”

Simulating Success

How do modeling and simulation activities, capabilities benefit Boeing? Let us count the ways—9 of them

By Debby Arkell

Hands-on experience often can be the best way to tackle complex problems or master challenging skills. But when it comes to navigating intricate, variable-laden scenarios, or combat situations involving complex military maneuvers using expensive equipment, "on-the-job training" often is not a prudent approach.

That's why Boeing Integrated Defense Systems, Commercial Airplanes and Phantom Works engage in a wide variety of modeling and simulation activities, designed to provide ever more realistic simulations to internal customers across the enterprise—and to external customers as well.

"There is a tremendous amount of diversity in modeling and simulation being worked on at Boeing, encompassing very complex issues within a very broad spectrum," said Ron Fuchs, director of Modeling and Simulation for IDS. "Right now there are more than

Boeing analysts have a variety of tools available—or under development—that can demonstrate concepts and provide significant cost savings by exploring ideas, developing systems, testing and manufacturing within a virtual environment before committing to specific approaches.

4 Product or system testing

Models and simulations are used to test

M&S used to test prototypes in variety of environments.

networked computer systems in the FCS system of systems will be tested in a large-scale distributed simulation facility called the FCS System of Systems Integration Lab. The SoSIL provides a

6 Training systems and maintenance

M&S are used to *train users in the operational environment* – enhancing learning. Simulation *costs 1/10* of running actual scenarios.

Network communications

Tactical military communications networks—such as Joint

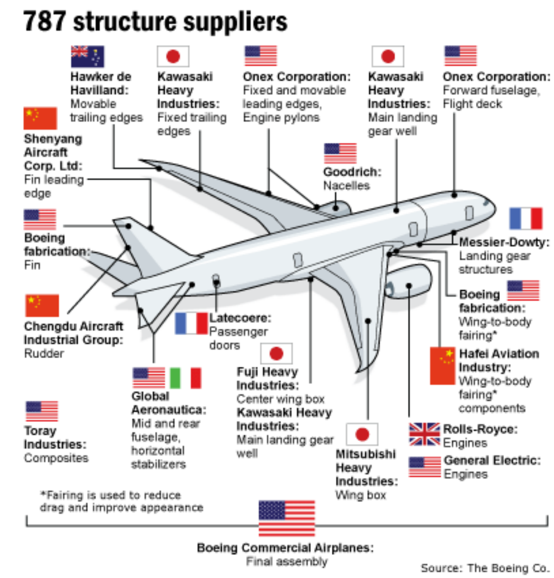
Scale of networks: cost-prohibitive or technically impossible for field tests.

M&S used to test and validate networking protocols in laboratory - environment acting as a test bed.

8

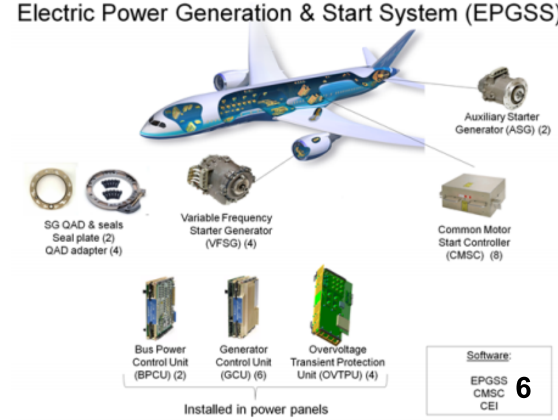
Calif., is an environment that acts as a distributed virtual test bed, and the Boeing Transformational Communications Laboratory in El Segundo, Calif., performs a similar function for satellite communications.

Large Number of Vendors for the Final System



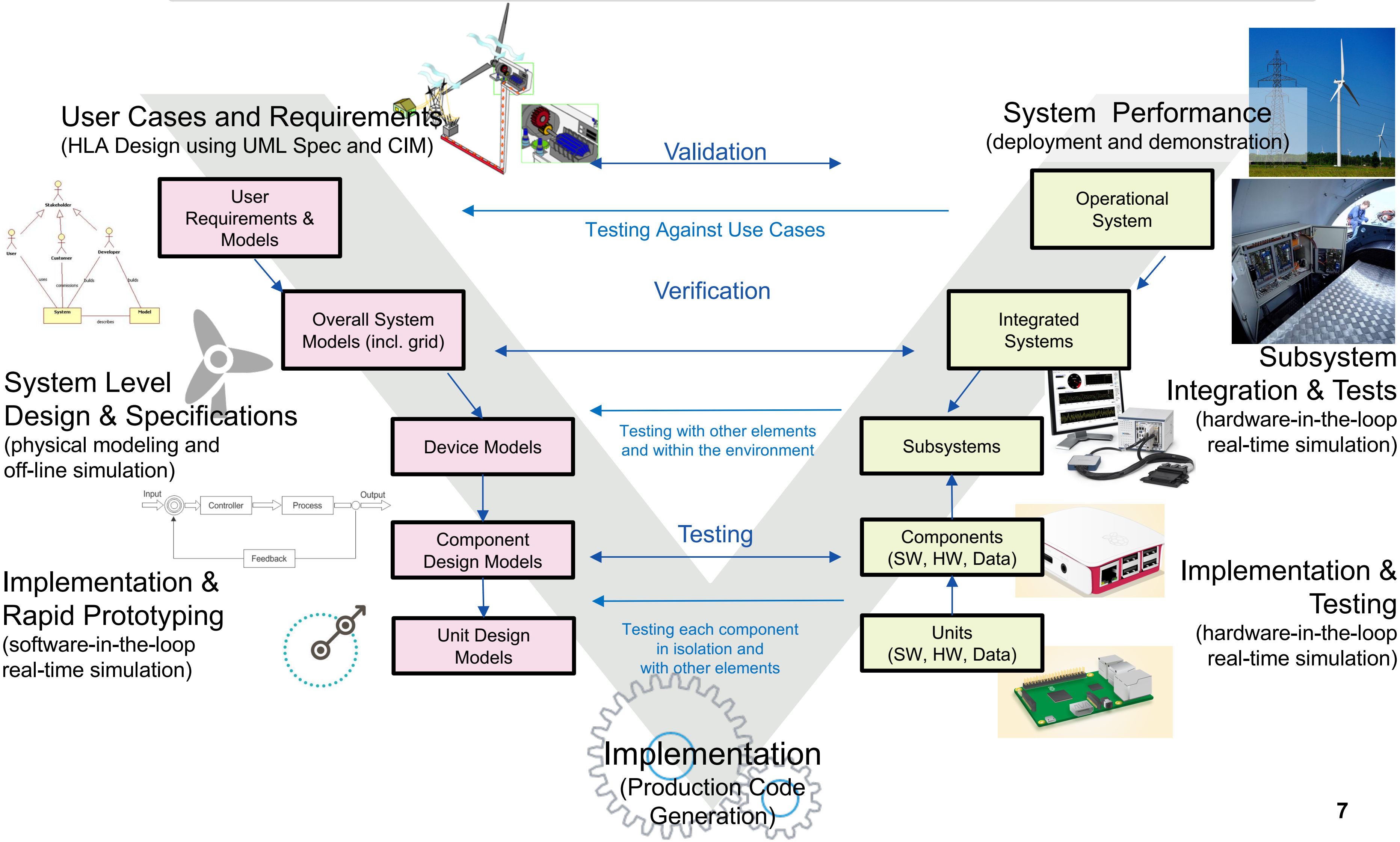
A Flying Micro-Grid!

Electric Power Generation & Start System (EPGSS)



The Multiple Roles of Modeling and Simulation
in building the future Cyber-Physical Power Systems (aka ‘smart grids’)

Conceptual Application for the Development of a WT Synchro phasor-Based Controller



Dangers of Models and Simulation

- **Falling in love with a model**
  The Pygmalion effect (forgetting that model is not the real world)
  - From the Greek myth of Pygmalion, a sculptor who fell in love with a statue he had carved.

- **Forcing reality into the constraints of a model**
  The Procrustes effect (e.g. economic theories)
  - Procrustes: "the stretcher [who hammers out the metal]", a rogue smith from Attica that physically attacked people by cutting/stretching their legs, so as to force them to fit the size of an iron bed.
  - A **Procrustean bed** is an **arbitrary standard** to which exact conformity is forced.

- **Forgetting the model's level of accuracy**
  Simplifying assumptions forgotten more than yesterday's pudding…

Dangers of Models and Simulation

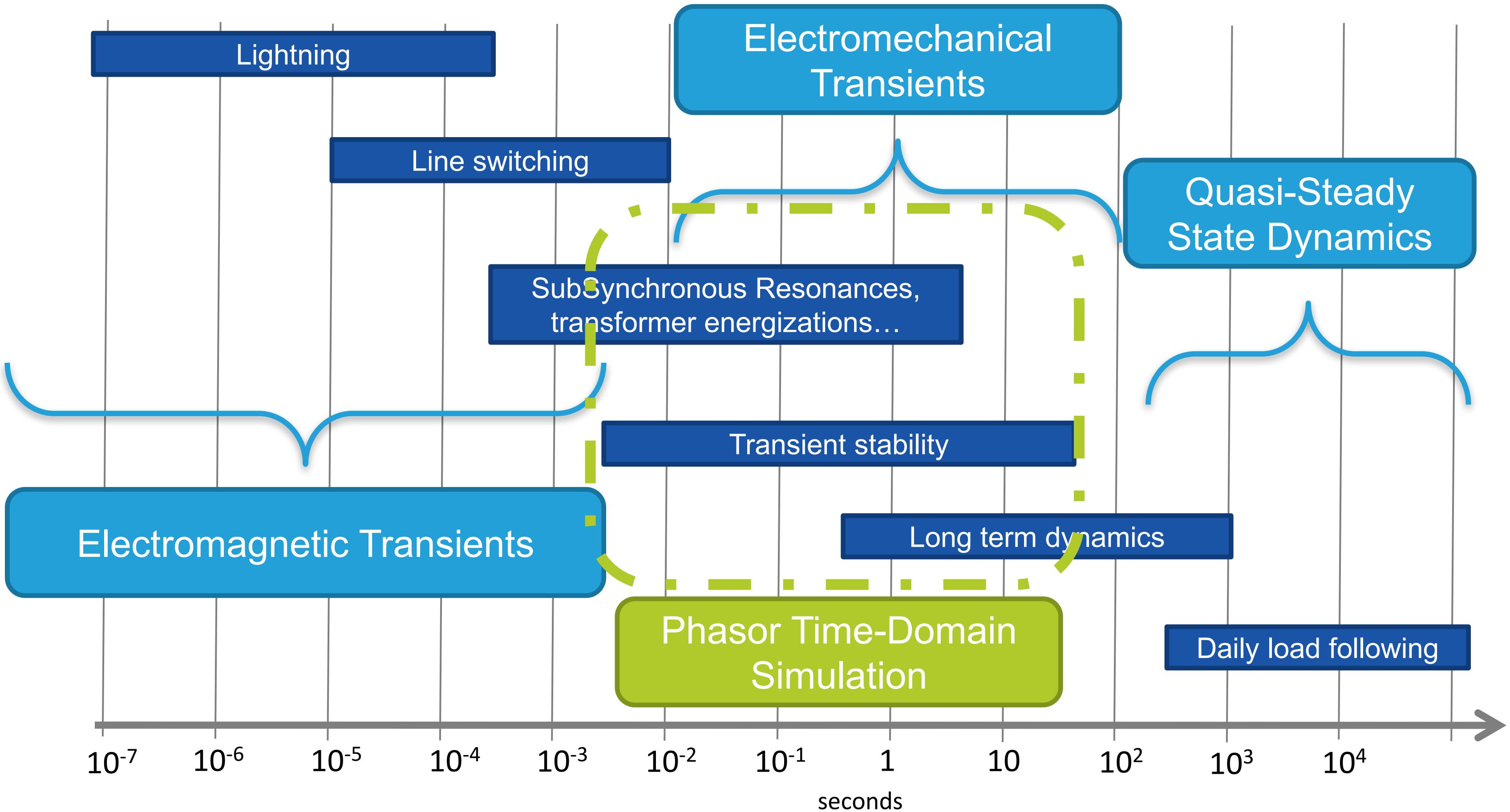
- **Falling in love with a model**
  The Pygmalion effect (forgetting that model is not the real world)
  - From the Greek myth of Pygmalion, a sculptor who fell in love with a statue he had carved.

- **Forcing reality into the constraints of a model**
  The Procrustes effect (e.g. economic theories)
  - Procrustes: "the stretcher [who hammers out the metal]", a rogue smith from Attica that physically attacked people by cutting/stretching their legs, so as to force them to fit the size of an iron bed.
  - A **Procrustean bed** is an **arbitrary standard** to which exact conformity is forced.

- **Forgetting the model's level of accuracy**
  Simplifying assumptions forgotten more than yesterday's pudding…

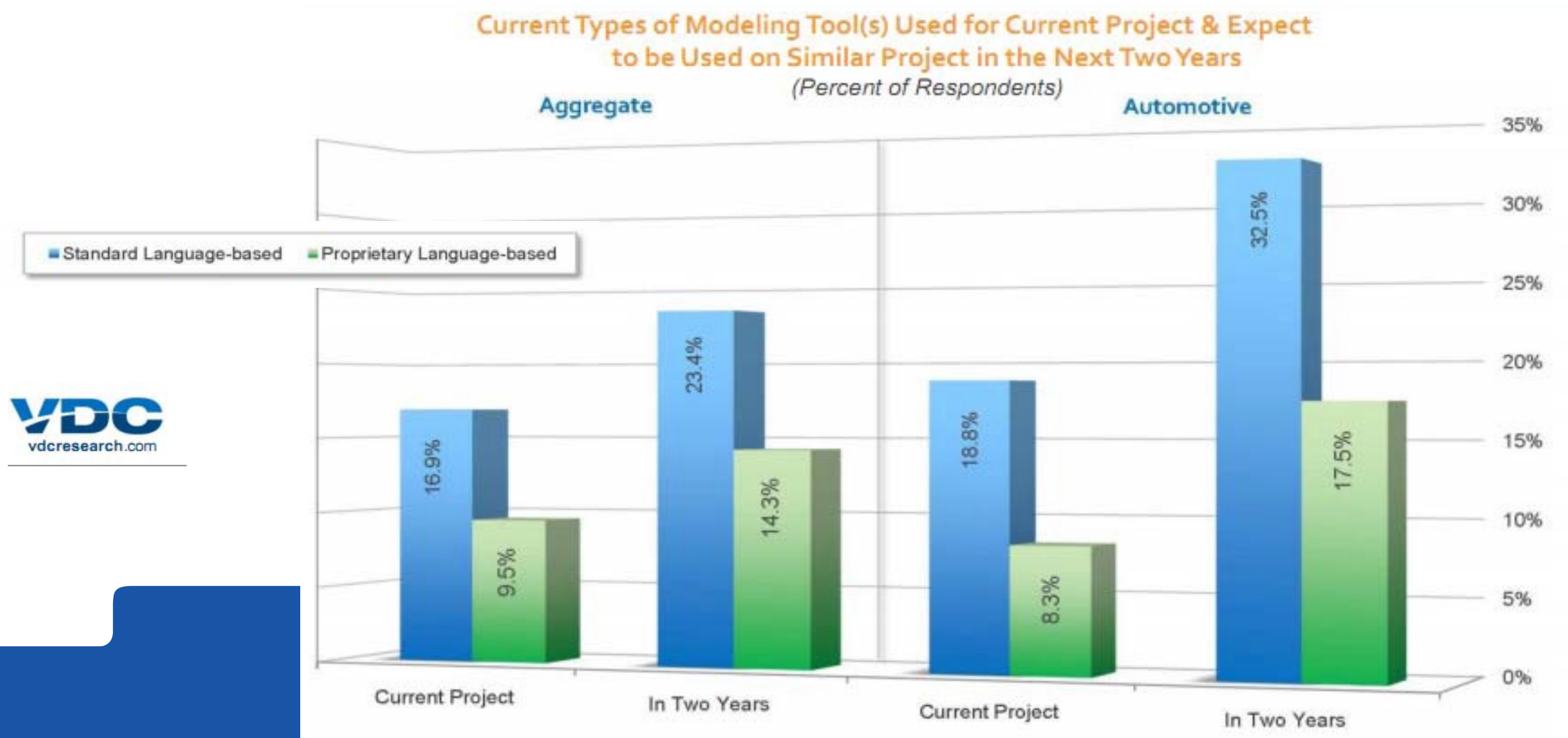
Phenomena modeled from this point on:

power system *electromechanical dynamics*

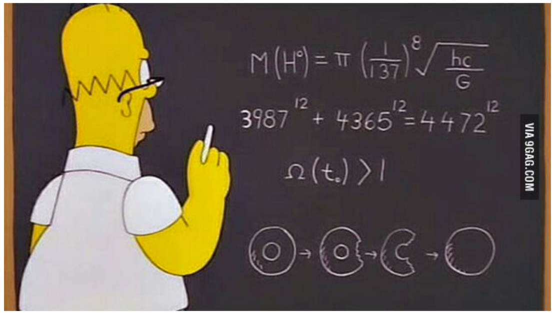


Why (open)-standardized modeling languages?

- Modeling tools first gained adoption as engineers looked for ways to simplify SW development and documentation.
- **Today's modeling tools and their use cases have evolved.**
- ***Now:*** need for addressing both system level design and SW development/construction.



Why equation-based modeling?



- Defines an implicit relation between variables.
  - **The data-flow between variables is defined right before simulation** of the model (not during the modelling process!)
- A system can be seen as a complete model or a set of individual components.
- **The user is (in principle) only concerned with the model creation, and does not have to deal with the underlying simulation engine (only if desired).**
- It also allows decomposing complex systems into simple sub-models easier to understand, share and reuse

is a (computer) language, is **not** a tool!

- Modelica is a free/*libre* object-oriented modeling language with a textual definition to describe physical systems using differential, algebraic and discrete equations.
- A Modelica modeling environment is needed to edit or to browse a Modelica model graphically in form of a composition diagram (= schematic).
- A Modelica translator is needed to transform a Modelica model into a form (usually C-code) which can be simulated by standard tools.
- A Modelica modeling and simulation environment provides both of the functionalities above, in addition to auxiliary features (e.g. plotting)

Modelica® - A Unified Object-Oriented Language for Systems Modeling

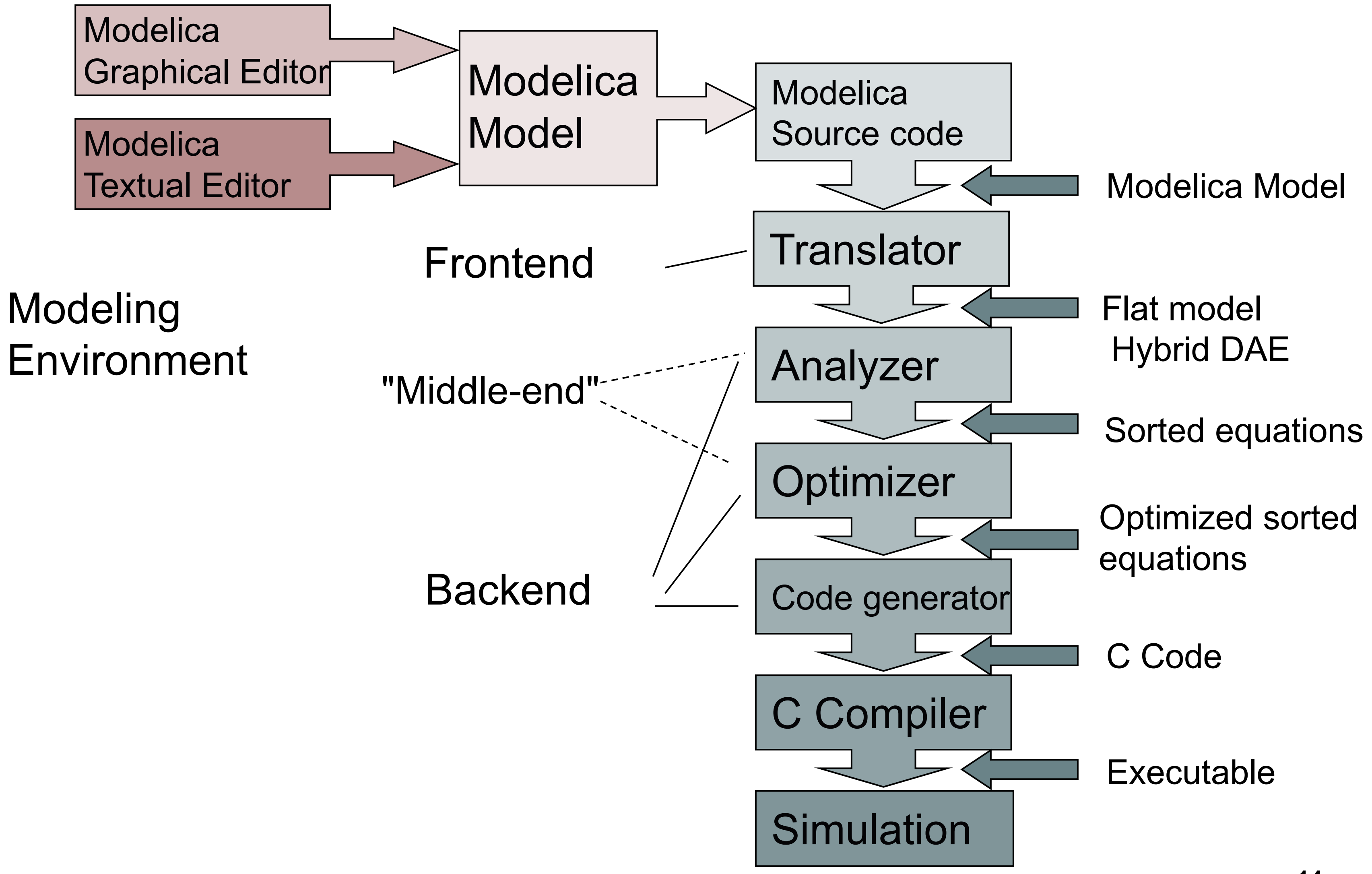
Language Specification

Version 3.3 Revision 1

July 11, 2014

Key: standardized and open language specification

Modelica modeling and simulation environment tasks



Why MODELICA power systems?

- The **order of computations** **is decided at modelling time**

Acausal	Causal

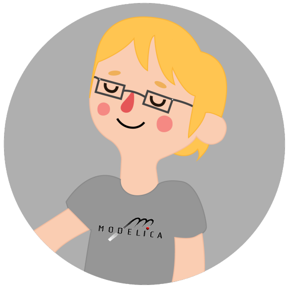
`R*I = v;`	`i := v/R;`
	`v := R*i;`
	`R := v/i;`

- Most tools make **no difference between “solver” and “model”** – in many cases solver is implanted in the model

- There is **no guarantee** that the same standardized model is implemented in the same way across different tools
- Even in Common Information Model (CIM) v15, **only block diagrams** are provided instead of equations

- Models are **black boxes** whose parameters are shared in a **specific “data format”**
- For large models this **requires translation** into the internal data format of each program

MODELICA and Power Systems



Previous and Related Efforts

- Modelica for power systems *was first attempted* in the early 2000's (Wiesmann & Bachmann, Modelica 2000) - "electro-magnetic transient (EMT) modeling" approach.
  - SPOT (Weissman, EPL-Modelon) and its close relative PowerSystems (Franke, 2014); supports multiple modeling approaches –i.e. 3phase, steady-state, "transient stability", etc.
- Electro-mechanical modeling or "transient stability" modeling:
  - Involves electro-mechanical dynamics, and neglects (very) fast transients
  - For system-wide analysis, easier to simulate/analyze - domain specific tools approach
- ObjectStab (Larsson, 2002; Winkler, 2015) adopts "transient stability" modeling.
- The PEGASE EU project (2011) developed a small library of components in Scilab, which where ported to proper Modelica in the FP7 iTesla project (2012-2016).
- The iPSL - iTesla Power Systems Library (Vanfretti et al, Modelica 2014, SoftwareX 2016), was released during 2015. Most models validated against typical power system tools.

OpenIPSL takes iPSL as a starting point and moves it forward (this presentation).

- F. Casella (OpenModelica 2016, Modelica 2017) presents the challenges of dealing with large power networks using Modelica, and a dedicated library to investigate them using the Open Modelica compiler.

MODELICA and Power Systems

Why another library for power systems?

- Why not use one of the existing Modelica projects?
  - *There is no technical argument:* in principle, either SPOT, PowerSystems, or ObjecStab could have been used instead of creating a new library (iPSL, and OpenIPSL)

Social Aspects (Vanfretti et al, Modelica 2014):

- Resistance to change: an irrational and dysfunctional reaction of users (and developers?)
  - Users of conventional power system tools are skeptical about any other tools different to the one they use (or develop), and are averse about new technologies (slow on the uptake)
- Change agents contribute (+/-) to address resistance through actions and interactions:
  - ***Strategy:*** do not impose the use of a specific simulation environment (software tool), *instead*,
  - Propose a **common human and computer-readable mathematical "description":** use of Modelica for unambiguous model exchange.
- Decrease of avoidance forces:
  - SW-to-SW validation gives quantitatively an similar answer than domain specific tools.
  - Accuracy (w.r.t. to *de* facto tools) more important than performance

A never-ending effort:

- Our (my) goal has been to bridge the gap between the Modelica and power systems community by
  - Addressing resistance to change (see above)
  - Interacting with both communities – different levels of success…

The *OpenIPSL* Project

- **KTH SmarTS Lab** (my research team) actively participated in the group or partners developing iPSL until the end of the *iTesla* project (March 2016)
- **iPSL** is a nice prototype, ***but we identified the following issues:***
  - **Development:** Need for compatibility with OpenModelica, (better) use of object orientation and `proper use of the Modelica language features.`
  - **Maintenance:** Poor harmonization, lack of code factorization, etc.
  - **Human issues:** The development workflow was complex, because of
    - Different parties with disparate objectives, levels of knowledge, philosophy, etc.

New research requirements and the experiences from previous effort indicated:

- a clear *need for a different development approach* –

one that should address a complex development & maintenance workflow!

- OpenIPSL *started as a **fork*** of iPSL
- OpenIPSL is hosted on GitHub at https://github.com/SmarTS-Lab/OpenIPSL
- OpenIPSL is actively developed by SmarTS Lab members and friends, as a research and education oriented library for power systems
  → ***it is ok to try things out*** !

Fork: copy of a project going *in a different development direction*

The *OpenIPSL* Library

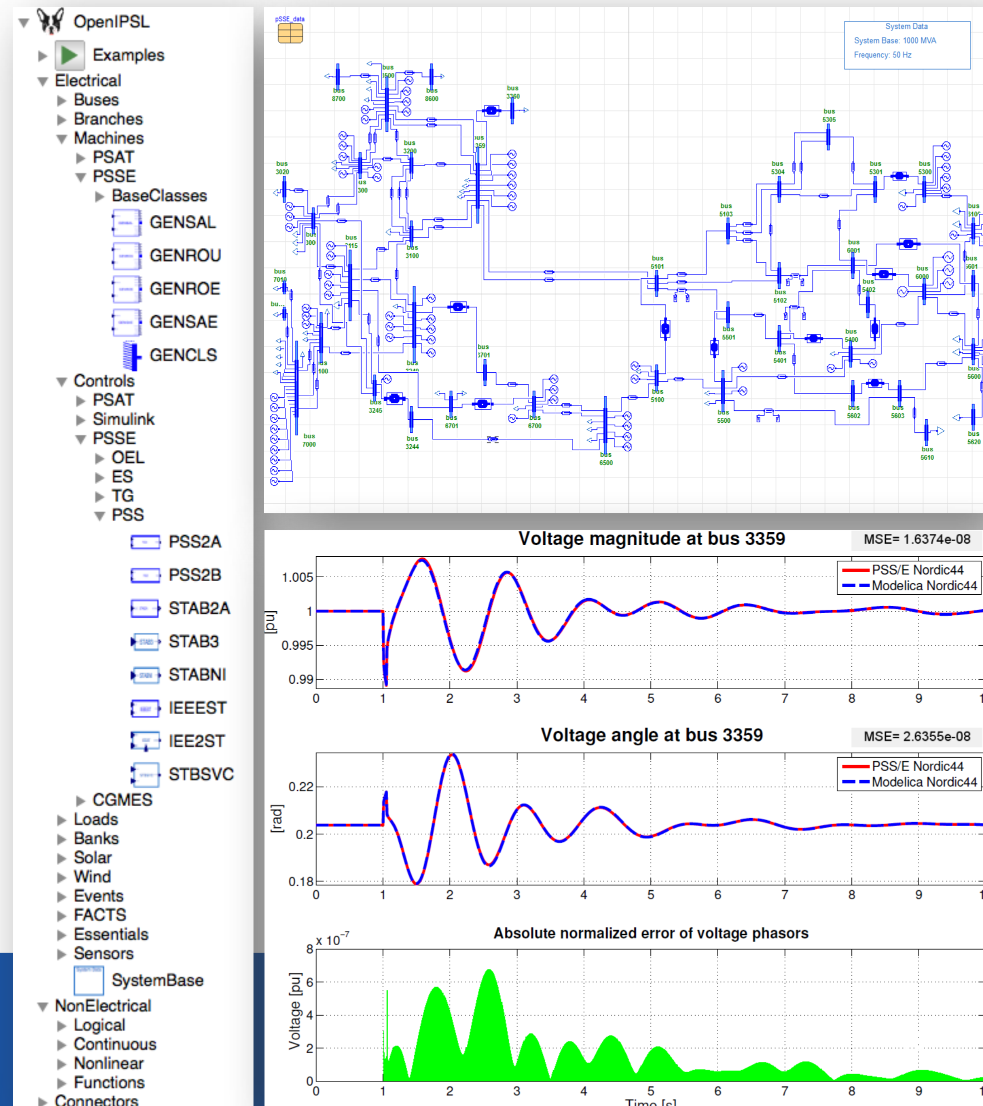
OpenIPSL

OpenIPSL is an open-source Modelica library for power systems

- It contains a set of **power system components** for **phasor time domain** modeling and simulation
- Models have been **validated** against a number of reference tools

OpenIPSL enables:

- **Unambiguous** model exchange
- Formal **mathematical description** of models
- **Separation** of **models** from IDEs and **solvers**
- Use of **object-oriented** paradigms

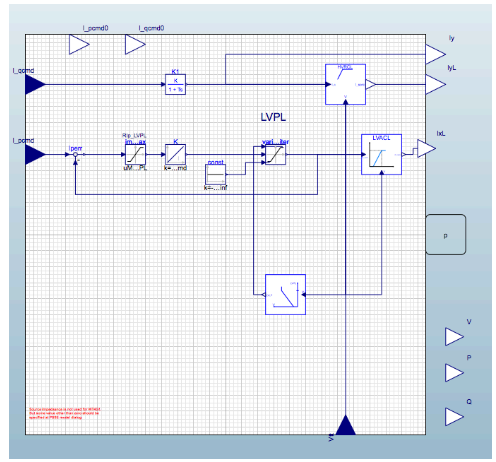


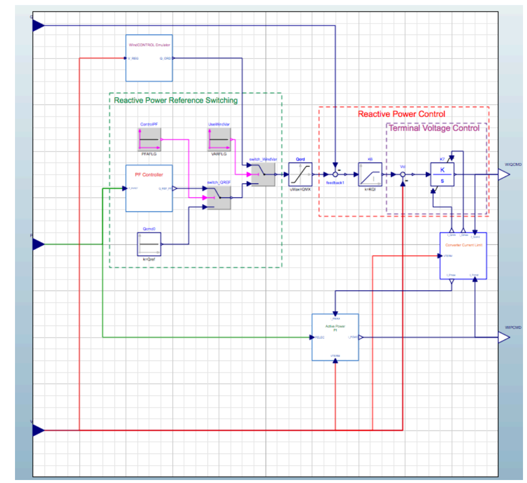
The *OpenIPSL* Library – WT Example

OpenIPSL

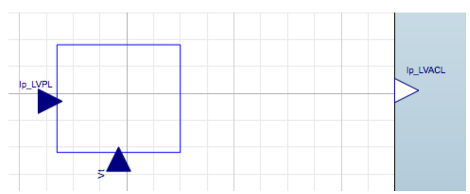
- OpenIPSL
  - Examples
  - Electrical
    - Buses
    - Branches
    - Machines
      - PSAT
      - PSSE
        - BaseClasses
        - GENSAL
        - GENROU
        - GENROE
        - GENSAE
        - GENCLS
    - Controls
      - PSAT
      - Simulink
      - PSSE
        - OEL
        - ES
        - TG
        - PSS
          - PSS2A
          - PSS2B
          - STAB2A
          - STAB3
          - STABNI
          - IEEEST
          - IEE2ST
          - STBSVC
      - CGMES
    - Loads
    - Banks
    - Solar
    - Wind
    - Events
    - FACTS
    - Essentials
    - Sensors
    - SystemBase
  - NonElectrical
    - Logical
    - Continuous
    - Nonlinear
    - Functions
  - Connectors

- Wind
  - PSAT
  - PSSE
    - WT3G
    - WT4G
      - WT4G1
      - WT4E1
      - Submodels
        - LVACL
        - HVRCL
        - LVPL
        - CCL





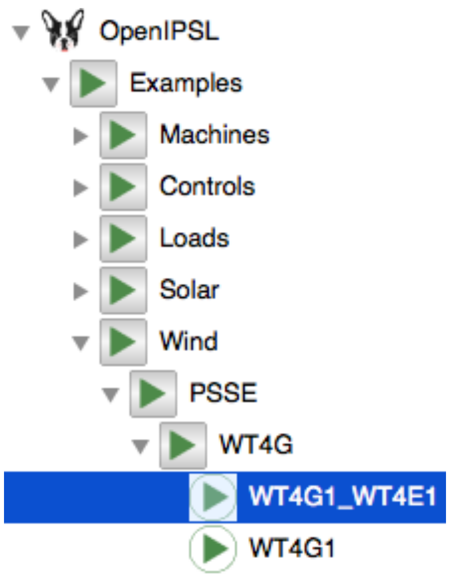
- Submodels
  - LVACL
  - HVRCL
  - LVPL
  - CCL

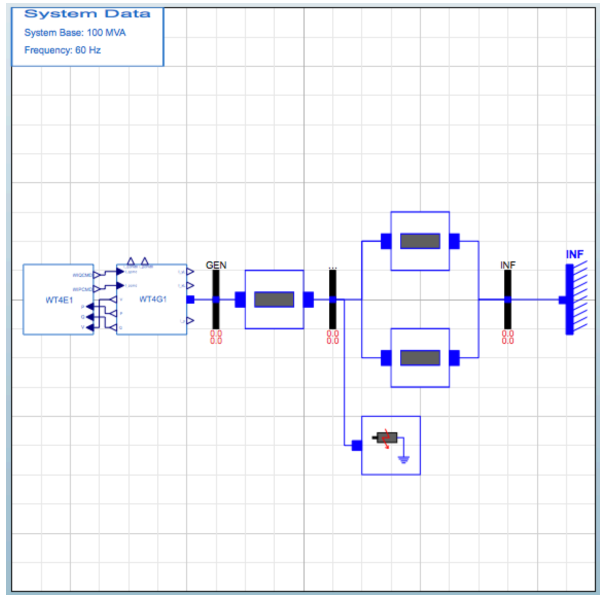



```
model LVACL
  //The Low Voltage Active Current Management block is de
  //of active power under very low voltage scenarios. Thi
  //  The protection function is activated when
  //the terminal voltage drops below 0.8 pu and stranglin
  //0.4 pu. For voltages between 0.8 pu and 0.4 pu to red
  Modelica.Blocks.Interfaces.RealOutput Ip_LVACL ;
  Modelica.Blocks.Interfaces.RealInput Vt ;
  Modelica.Blocks.Interfaces.RealInput Ip_LVPL ;
equation
  if Vt < 0.4 then
    Ip_LVACL = 0;
  elseif Vt > 0.8 then
    Ip_LVACL = Ip_LVPL;
  else
    Ip_LVACL = Ip_LVPL * 1.25 * Vt;
  end if;
  ;
end LVACL;
```

The *OpenIPSL* Library – Network Example

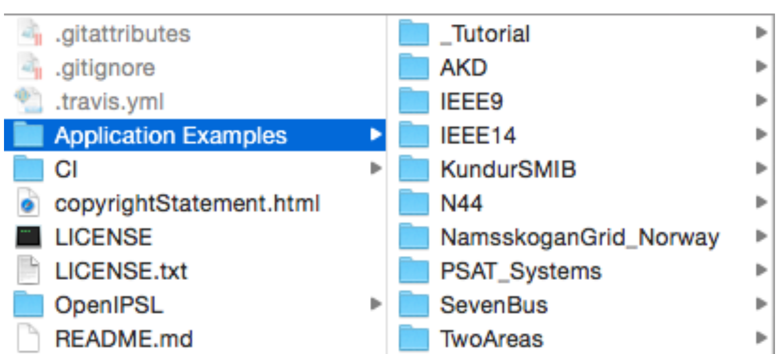
OpenIPSL






```
model WT4G1_WT4E1
  extends Modelica.Icons.Example;
  constant Real pi = Modelica.Constants.pi;
  parameter Real V1 = 1.0;
  parameter Real A1 = -1.570655e-05;
  parameter Real V3 = 0.9999999000000001;
  parameter Real A3 = 0.02574992;
  parameter Real P1 = -1.4988;
  parameter Real Q1 = -4.334;
  parameter Real Zr = 0.0;
  parameter Real Zi = 0.2;
  parameter Real P3 = 1.5;
  parameter Real Q3 = -5.6658;
  parameter Real R1 = 0.025;
  parameter Real X1 = 0.025;
  parameter Real B1 = 0.05;
  parameter Real dyrw[1, 9] = [0.02, 0.02,
  OpenIPSL.Electrical.Branches.PwLine pwLin
  OpenIPSL.Electrical.Branches.PwLine pwLin
  OpenIPSL.Electrical.Machines.PSSE.GENCLS
  OpenIPSL.Electrical.Branches.PwLine pwLin
  OpenIPSL.Electrical.Wind.PSSE.WT4G.WT4G1
  OpenIPSL.Electrical.Events.PwFault pwFaul
  OpenIPSL.Electrical.Wind.PSSE.WT4G.WT4E1
  inner OpenIPSL.Electrical.SystemBase SysD
  OpenIPSL.Electrical.Buses.Bus GEN ;
  OpenIPSL.Electrical.Buses.Bus BUS1 ;
  OpenIPSL.Electrical.Buses.Bus INF ;
equation
  connect(wT4G1.p, GEN.p) ;
  connect(GEN.p, pwLine2.p) ;
  connect(pwLine2.n, BUS1.p) ;
  connect(BUS1.p, pwLine.p) ;
  connect(pwLine1.p, pwLine.p) ;
  connect(pwFault.p, BUS1.p) ;
  connect(pwLine.n, INF.p) ;
  connect(pwLine1.n, INF.p) ;
  connect(INF.p, gENCLS2_1.p) ;
  connect(wT4E1_1.WIQCMD, wT4G1.I_qcmd) ;
  connect(wT4E1_1.WIPCMD, wT4G1.I_pcmd) ;
  connect(wT4G1.P, wT4E1_1.P) ;
  connect(wT4G1.V, wT4E1_1.V) ;
  connect(wT4G1.Q, wT4E1_1.Q) ;
  ;
end WT4G1_WT4E1;
```

Many Application Examples Developed!!!



Initialization (1/3) - General DAE Model

$$\dot{\mathbf{x}} = f\left(\mathbf{x}, \mathbf{y}, \boldsymbol{\eta}, \tilde{\mathbf{u}}, t\right),$$

$$\mathbf{0} = g\left(\mathbf{x}, \mathbf{y}, \boldsymbol{\eta}, \tilde{\mathbf{u}}, t\right).$$

$\mathbf{x}$ – is the vector of state variables, $\mathbf{x} = \tilde{\boldsymbol{\xi}}_i$

$\mathbf{y}$ – is the vector of algebraic variables, $\mathbf{y} = \tilde{\boldsymbol{\xi}}_f$

$\boldsymbol{\eta}$ – is the vector of parameters, from discarding $\tilde{\varphi}_s$ and letting $\tilde{\boldsymbol{\xi}}_s = \boldsymbol{\eta}$

$\tilde{\mathbf{u}}$ – is the vector of discrete variables.

$f(\cdot)$ – are the differential equations, $f(\cdot) \equiv \tilde{\varphi}_i(\cdot)$

$g(\cdot)$ – are the algebraic equations, $g(\cdot) \equiv \tilde{\varphi}_f(\cdot)$

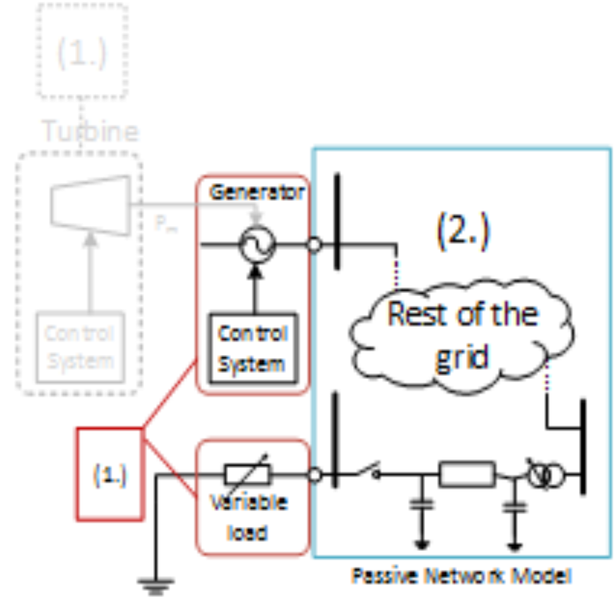
[Ref.] F. Milano, Power System Modeling and Scripting, Springer, 2010.

Initialization (2/3) - Power System Approach

- Equation set ***g*** is separated in two sets of algebraic equations:

$$\left.\begin{aligned}\dot{\mathbf{x}} &= f\left(\mathbf{x},\mathbf{y},\boldsymbol{\eta},\tilde{\mathbf{u}}\right),\\ \mathbf{0} &= g_1\left(\mathbf{x},\mathbf{y},\boldsymbol{\eta},\tilde{\mathbf{u}}\right),\end{aligned}\right\} \quad (1.)$$

$$\mathbf{0} = g_2\left(\mathbf{y},\boldsymbol{\eta},\tilde{\mathbf{u}}\right)\} \quad (2.)$$



(1) Is the part which governs how dynamic models will evolve, since they depend on both **x** and **y** , e.g. generators and their control systems.

(2) Is the network model, consisting of transmission lines and other passive components which only depends on algebraic variables, **y**

Initialization (3/3) – Differences

- The power system needs to be at rest, i.e. its states must have converged to a fixed point before a disturbance is applied in simulation, that is **x(0) = C**
  - Q: How can we find this equilibrium for a DAE system?
  - A: Set derivatives to zero and solve for all unknown variables!

$$\mathbf{0} = f\left(\mathbf{x}, \mathbf{y}, \boldsymbol{\eta}, \tilde{\mathbf{u}}, t\right),$$
$$\mathbf{0} = g\left(\mathbf{x}, \mathbf{y}, \boldsymbol{\eta}, \tilde{\mathbf{u}}, t\right).$$

Modelica –compliant tools attempt to solve this problem!

- Some observations that can be made:
  - The algebraic equations in corresponded to having the fast differential equations at equilibrium all the time (in the model and in the timescale considered).
  - Finding the equilibrium when most of the variables are unknown is very difficult if when we try to solve this equation system simultaneously.
- Power system tools do not do this (to the best of my knowledge)!
  - In power systems, we attempt to sequentially solve the equation system at *t=0.*
  - First, we need to solve the algebraic equations *g* that only depend on the algebraic variables… and then solve *f=0.*

The *OpenIPSL* – “initial guess” approach

OpenIPSL

- **An initial guess for all algebraic, continuous and discrete variables need to be provided to solve a `numerical problem!`**
- When solving differential equations, one needs to provide the **initial value** of the state variables at rest.

- In Modelica, **initial values can be either solved or specified** in many ways, we use the following
  - Using the ”initial equation” construct:
```
initial equation
    x = some_value OR x = expression to solve
```
  - Setting the (*`fixed=true, start=x0`*) attribute when instantiating a model, will
  - If nothing is specified, the default would be a guess value (*`start= 0, fixed=false`*).

- In the OpenIPSL models we do the following:
  - The initial guess value is set with (*`fixed = false`*) for initialization.
  - Model attributes are treated as parameters with value (*`fixed = true`*),

- In OpenIPSL we use a power flow solution from an external tool (e.g. PSAT or PSS/E) as a starting point to compute initial guess values through parameters within each model.
  - The power flow solution is NOT the initial guess value itself.
  - Aim is to provide a better “initial guess” to find the initial values of the DAE system.

The *OpenIPSL* – “initial guess” example

Third order model from PSAT implemented in OpenIPSL

OpenIPSL

Power flow data

V_b	400
V_0	1
angle_0	0
P_0	1
Q_0	0
S_b	SysData.S_b
fn	SysData.fn

Base voltage of the bus (kV)

Initialization

w	1		Rotor speed (pu)
v	V_0	true	Generator terminal voltage (pu)
P	P_0 / S_b		Active power (pu)
Q	Q_0 / S_b		Reactive power (pu)
anglev	angle_0 / 180 * pi		Bus voltage angle
e1q	e1q0		q-axis transient voltage (pu)

```
model Order3 "Third Order Synchronous Machine with Inputs and Outputs"
  import Modelica.Constants.pi;
  extends BaseClasses.baseMachine(delta(start = delta0), pe(start = pm00), pm(start = pm00)

  Real e1q(start = e1q0) "q-axis transient voltage (pu)";
protected
  parameter Real Xd = xd * CoB "d-axis reactance, p.u.";
  parameter Real x1d = xd1 * CoB "d-axis transient reactance, p.u.";
  parameter Real Xq = xq * CoB "q-axis reactance, p.u.";
  parameter Real m = M / CoB2 "Machanical starting time (2H), kWs/kVA";
  parameter Real c1 = Ra * K "CONSTANT";
  parameter Real c2 = x1d * K "CONSTANT";
  parameter Real c3 = Xq * K " CONSTANT";
  parameter Real K = 1 / (Ra * Ra + Xq * x1d) "CONSTANT";
  parameter Real delta0 = atan2(vi0 + Ra * ii0 + Xq * ir0, vr0 + Ra * ir0 - Xq * ii0) "Initialitation";
  parameter Real vd0 = vr0 * cos(pi / 2 - delta0) - vi0 * sin(pi / 2 - delta0) "Initialitation";
  parameter Real vq0 = vr0 * sin(pi / 2 - delta0) + vi0 * cos(pi / 2 - delta0) "Initialitation";
  parameter Real id0 = ir0 * cos(pi / 2 - delta0) - ii0 * sin(pi / 2 - delta0) "Initialitation";
  parameter Real iq0 = ir0 * sin(pi / 2 - delta0) + ii0 * cos(pi / 2 - delta0) "Initialitation";
  parameter Real pm00 = (vq0 + Ra * iq0) * iq0 + (vd0 + Ra * id0) * id0 "Initialitation";
  parameter Real vf00 = e1q0 + (Xd - x1d) * id0 "Initialitation";
  parameter Real e1q0 = vq0 + Ra * iq0 + x1d * id0 "Initialitation";
initial equation
  der(e1q) = 0;
equation
  der(e1q) = ((-e1q) - (Xd - x1d) * id + vf) / Td10;
```

The *OpenIPSL* Project Documentation

OpenIPSL

The intention is to have comprehensive documentation in the repositories:

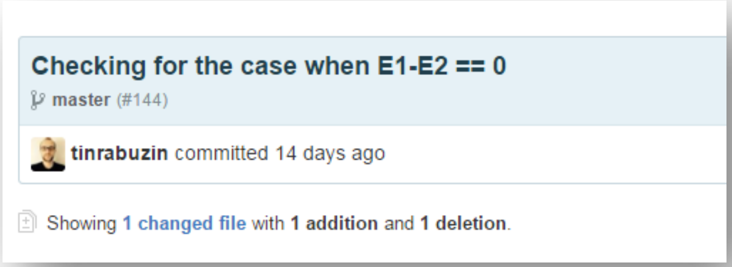
- Documentation of the code changes

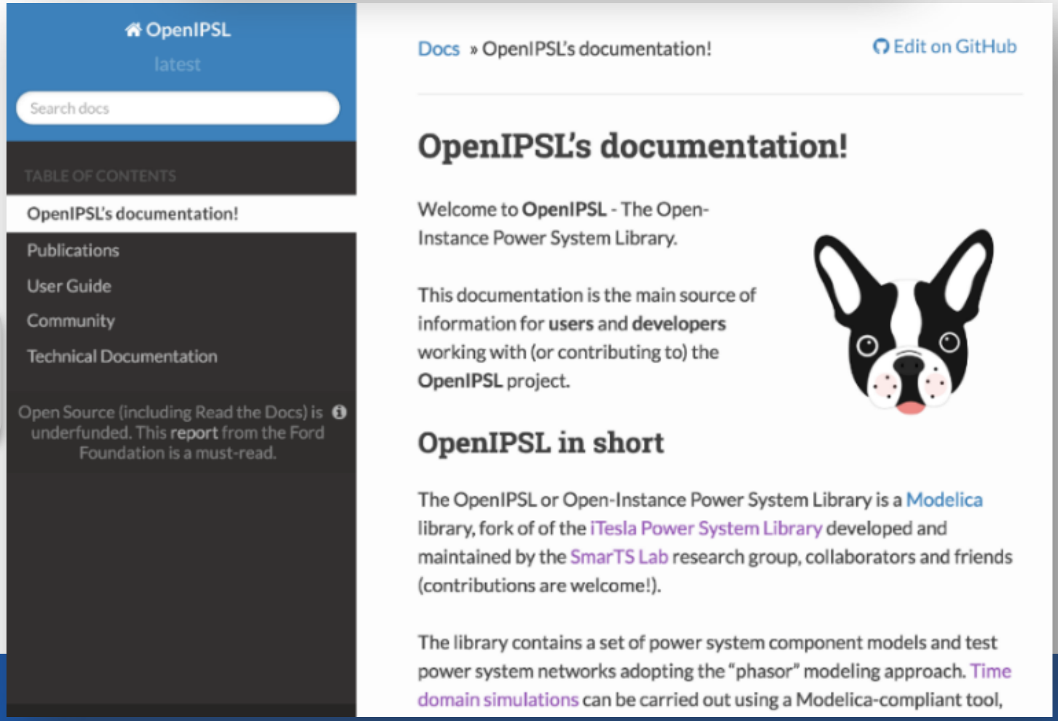
→ Explicit messages in ***commits*** and ***pull-requests***

- Documentation of the project
  - Presentation
  - User guide
  - Dev. guidelines & How to contribute

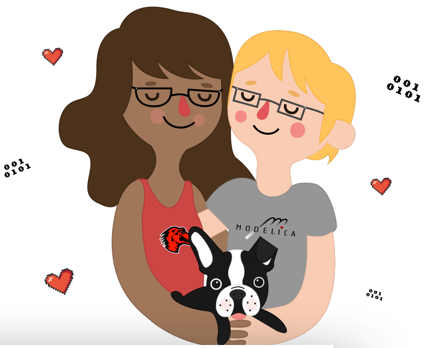
→ The documentation is written in ***reStructuredText*** (reST) hosted on http://openipsl.readthedocs.io/

Note: Model documentation is not included, users are referred to the proprietary documentations.





The *OpenIPSL* Project
Latest Developments/Contributions



Some of the latest development in the library:

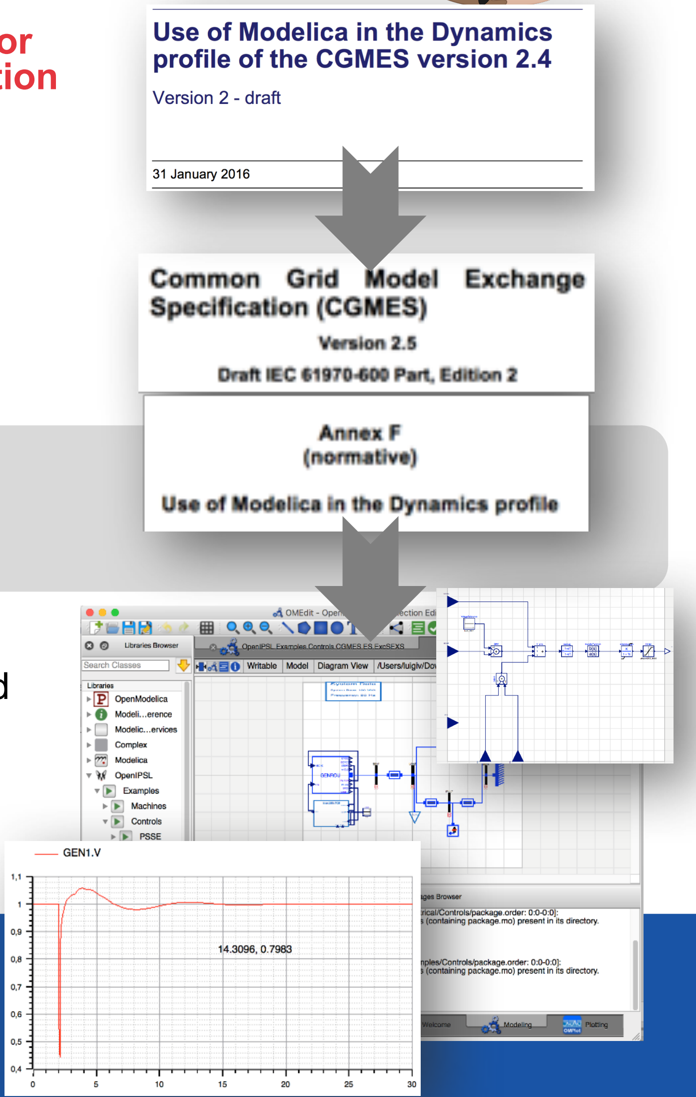
- **100% Compatibility with OM (100% Check, 100% Simulation for components) through efforts in Continuous Integration adoption**
- Change in the models to include inheritance (code factorizing)
- Fixing and validating network models (thanks to CI)
- Component for interfacing OpenIPSL with 3 phase models (aka MonoTri)
  - For distribution grid (unbalanced) simulations
  - Starting point for mixed transmission and distribution network simulations

ENTSO-E IOP:

- Proof of concept and test model
- Excitation system and small network model

OpenCPS Models

- Small power network models for analysis of continuous and hybrid systems (sampling and discretized AVR model)
- Process noise (gen./load) **pdf-based** load models added
- Frequency estimation model
- Sequential automated re-synchronization and control model for islanded network



New research requirements and the experiences from previous effort indicated ***a clear need for a different development approach*** **- one** that should address a complex development and maintenance workflow!

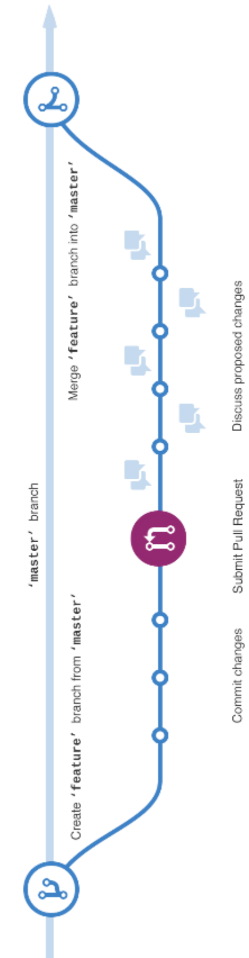
How to master a complex development workflow?

Continuous Integration

A Collaborative Workflow

We adopted the ***pull-request*** workflow (or GitHub workflow):

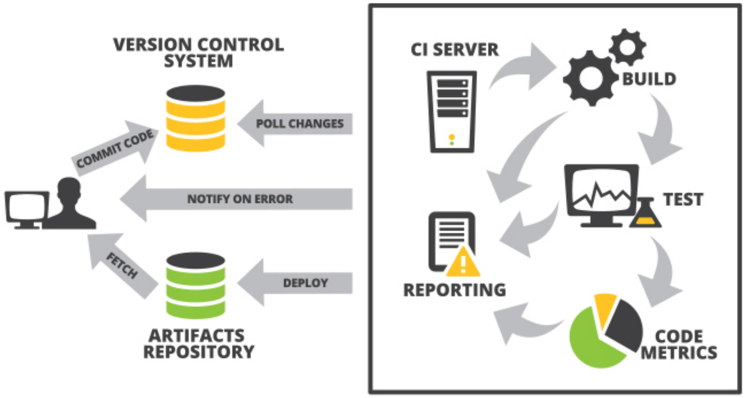
- Participants ***fork*** the repository and work in their repository
- Changes are submitted to the main repository as ***pull-requests***
- The pull-requests are ***reviewed*** by “admin” members of the repository
  - upon ***validation*** the changes are merged in the code of the repository



- Mistakes can be made by members of our team, ***we are still learning!***
- The Git workflow adopted allows to minimize the impact of these errors.
- Increased library quality!

Toward Continuous Integration

- The ***previous workflow*** was used by only ***few people*** and resulted in ***no control*** over the code quality, *even though DVCS was being used.*
- The ***newly adopted*** workflow turned suitable for the development ***team***, but generated a strong ***burden*** for the ***code review***



This sparked the idea of implementing a ***Continuous Integration workflow:***

→ Focus on "***lighter***", ***more frequent*** pull-requests, containing ***less code*** change, all related to a ***single feature*** to facilitate the code validation

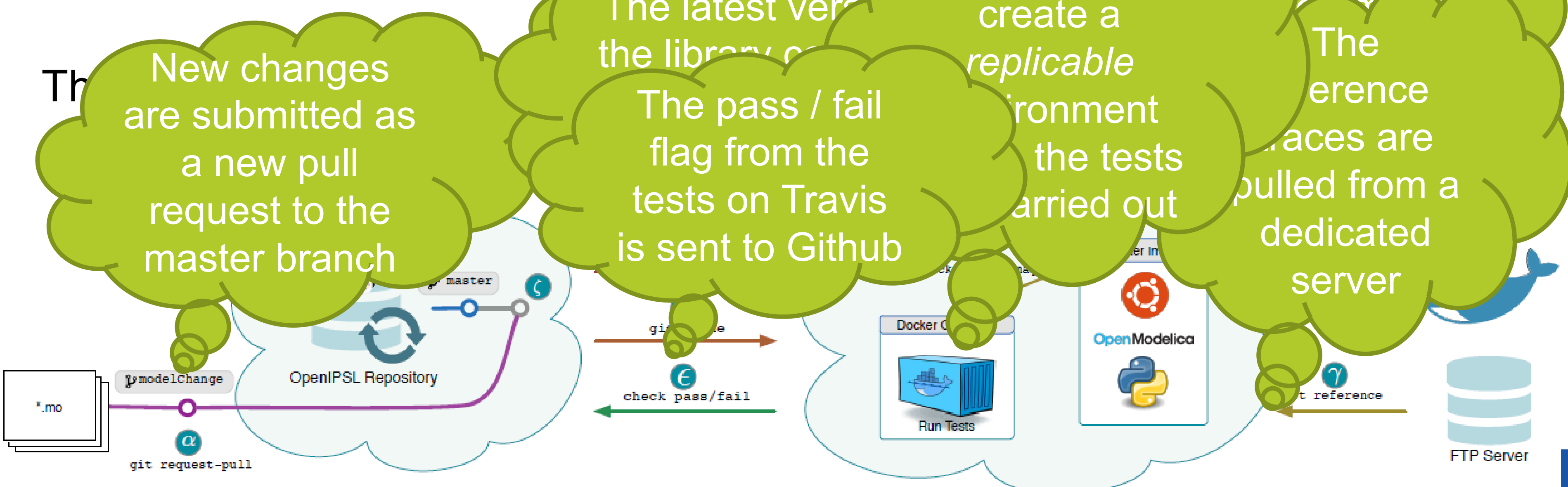
→ Implement a CI service to ***automate*** recurring code ***validation tests***, to liberate "admin" resources.

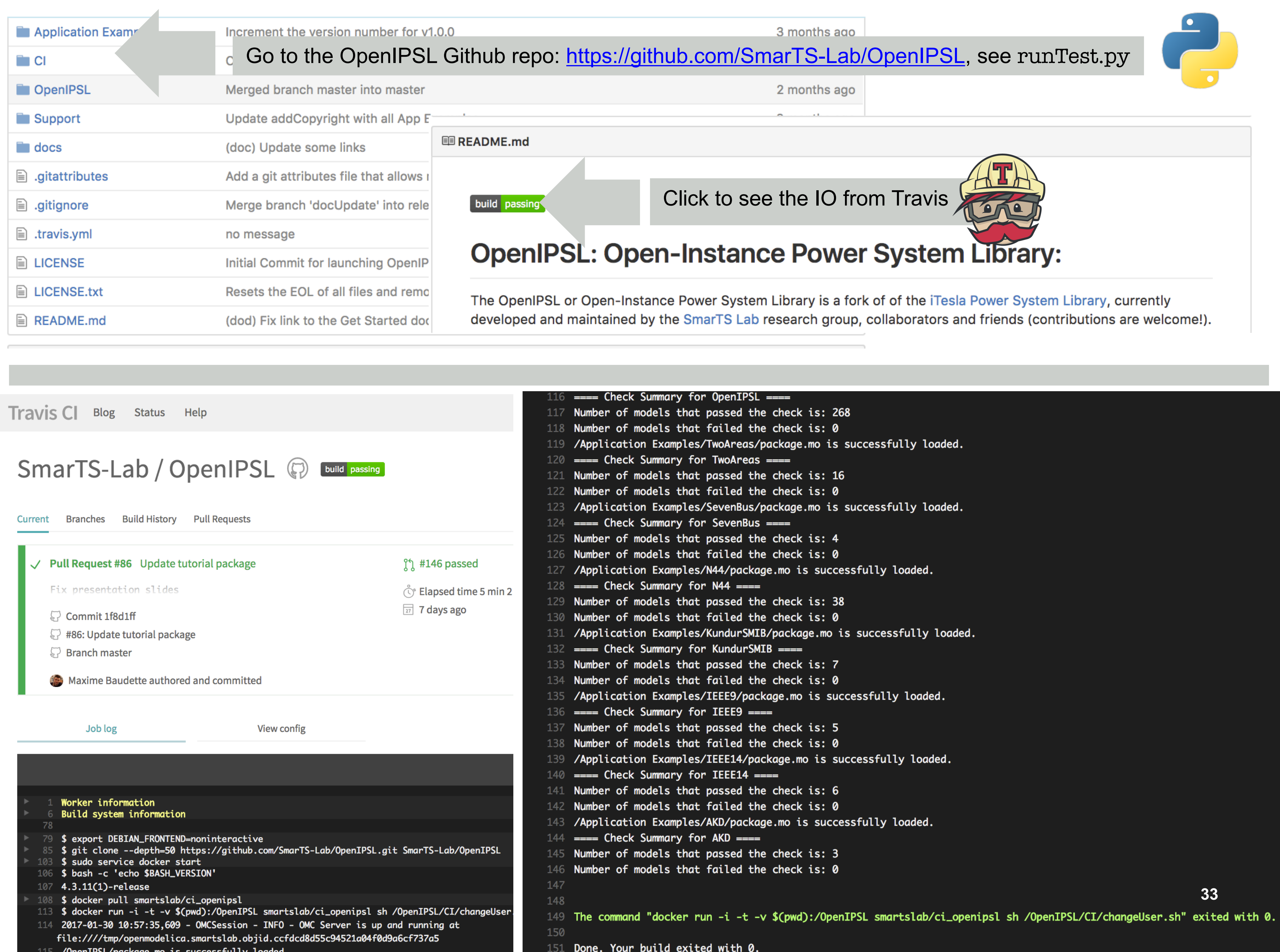
Continuous Integration (CI) Service

A CI service was implemented and integrated to the repository. The Modelica support was achieved with the following architecture:

- ***Travis*** as CI service provider
- ***Docker*** as the “virtualization” architecture
- ***DockerHub*** to host a Do

Th





Go to the OpenIPSL Github repo: https://github.com/SmarTS-Lab/OpenIPSL, see runTest.py
Application Exam | Increment the version number for v1.0.0 | 3 months ago
CI
OpenIPSL | Merged branch master into master | 2 months ago
Support | Update addCopyright with all App E
docs | (doc) Update some links
.gitattributes | Add a git attributes file that allows
.gitignore | Merge branch 'docUpdate' into rele
.travis.yml | no message
LICENSE | Initial Commit for launching OpenIP
LICENSE.txt | Resets the EOL of all files and remo
README.md | (dod) Fix link to the Get Started doc
README.md
build passing
Click to see the IO from Travis
OpenIPSL: Open-Instance Power System Library:
The OpenIPSL or Open-Instance Power System Library is a fork of of the iTesla Power System Library, currently developed and maintained by the SmarTS Lab research group, collaborators and friends (contributions are welcome!).
Travis CI Blog Status Help
SmarTS-Lab / OpenIPSL build passing
Current Branches Build History Pull Requests
Pull Request #86 Update tutorial package
Fix presentation slides
#146 passed
Elapsed time 5 min 2
7 days ago
Commit 1f8d1ff
#86: Update tutorial package
Branch master
Maxime Baudette authored and committed
Job log View config
Worker information
Build system information
$ export DEBIAN_FRONTEND=noninteractive
$ git clone --depth=50 https://github.com/SmarTS-Lab/OpenIPSL.git SmarTS-Lab/OpenIPSL
$ sudo service docker start
$ bash -c 'echo $BASH_VERSION'
4.3.11(1)-release
$ docker pull smartslab/ci_openipsl
$ docker run -i -t -v $(pwd):/OpenIPSL smartslab/ci_openipsl sh /OpenIPSL/CI/changeUser
2017-01-30 10:57:35,609 - OMCSession - INFO - OMC Server is up and running at file:////tmp/openmodelica.smartslab.objid.ccfdcd8d55c94521a04f0d9a6cf737a5
/OpenIPSL/package.mo is successfully loaded.
==== Check Summary for OpenIPSL ====
Number of models that passed the check is: 268
Number of models that failed the check is: 0
/Application Examples/TwoAreas/package.mo is successfully loaded.
==== Check Summary for TwoAreas ====
Number of models that passed the check is: 16
Number of models that failed the check is: 0
/Application Examples/SevenBus/package.mo is successfully loaded.
==== Check Summary for SevenBus ====
Number of models that passed the check is: 4
Number of models that failed the check is: 0
/Application Examples/N44/package.mo is successfully loaded.
==== Check Summary for N44 ====
Number of models that passed the check is: 38
Number of models that failed the check is: 0
/Application Examples/KundurSMIB/package.mo is successfully loaded.
==== Check Summary for KundurSMIB ====
Number of models that passed the check is: 7
Number of models that failed the check is: 0
/Application Examples/IEEE9/package.mo is successfully loaded.
==== Check Summary for IEEE9 ====
Number of models that passed the check is: 5
Number of models that failed the check is: 0
/Application Examples/IEEE14/package.mo is successfully loaded.
==== Check Summary for IEEE14 ====
Number of models that passed the check is: 6
Number of models that failed the check is: 0
/Application Examples/AKD/package.mo is successfully loaded.
==== Check Summary for AKD ====
Number of models that passed the check is: 3
Number of models that failed the check is: 0
The command "docker run -i -t -v $(pwd):/OpenIPSL smartslab/ci_openipsl sh /OpenIPSL/CI/changeUser.sh" exited with 0.
Done. Your build exited with 0.

Extension of the CI Service

The ***first implementation*** eliminated parts of the '*rebarbative'* tasks by automating the ***code checks***:

- Avoid error propagation in the library, models "out-of-sync"
- Implementation entirely based on ***OpenModelica***
  → ***100% OM Compatibility*** achieved !

From this successful implementation, an extension was investigated to ***include model validation*** into the CI service:

- Model validation tests were carried out "offline" during the model development stages
  → ***We did it before***!
- Automated model validation (aka regression testing), ensures code changes won't affect existing models
  → Library ***integrity guaranteed***

Model Validation Workflow (SW-to-SW) (1/2)

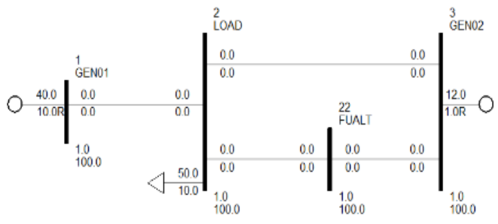
In the original implementation of the models of the OpenIPSL, a software-to-software validation workflow was designed and carried out "offline":

- Models are implemented from several ***reference programs***
  - ***PSAT***, domain specific tool in MATLAB/Simulink by F. Milano
  - ***PSS/E***, domain specific tool from Siemens PTI
- Modelica models were validated using ***small scale*** power network
- The traces from the Modelica models were qualitatively and quantitatively assessed: compared to the ***reference traces***

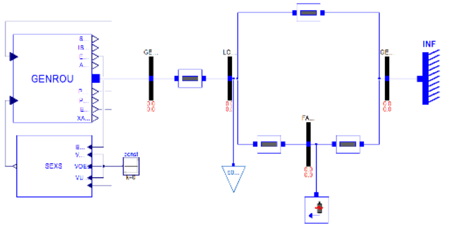
→ Gives ***confidence*** to users having a long experience with these reference software …

Model Validation Workflow (SW-to-SW) (2/2)

Reference SW Tool



Modelica Model



Power Flow Calculations

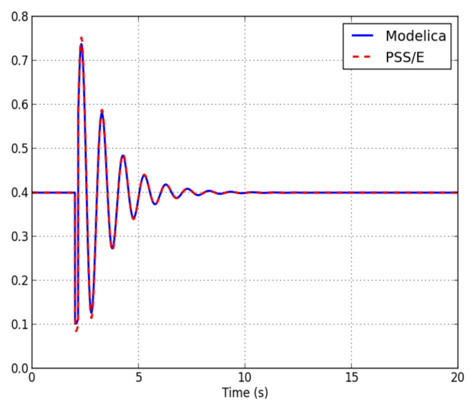
Time-domain simulation

Time-domain simulation

Graphical and Quantitative Assessment

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

Signal P



Model	Validation Result
EXAC1	Fail
EXAC2	Fail
EXST1	Pass

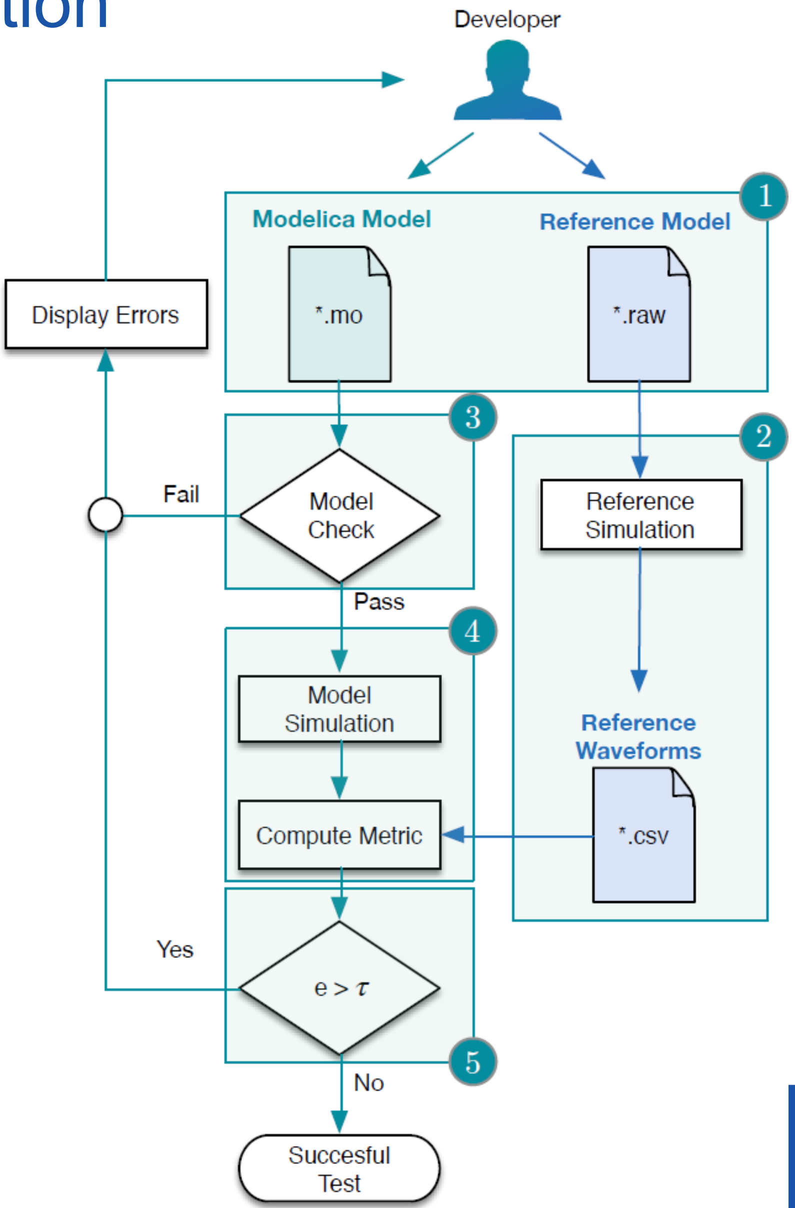
Signal	RMSE	Plot
P	2.8389e-3	+

Continuous Integration (CI)
Full workflow implementation

Workflow Summary:

- A two-stage process
  - Modelica ***syntax*** check
  - Model ***validation*** check
- Fully automated through online CI services

→ Diagnostic help to the developers to ***locate the error***



Continuous Integration (CI)

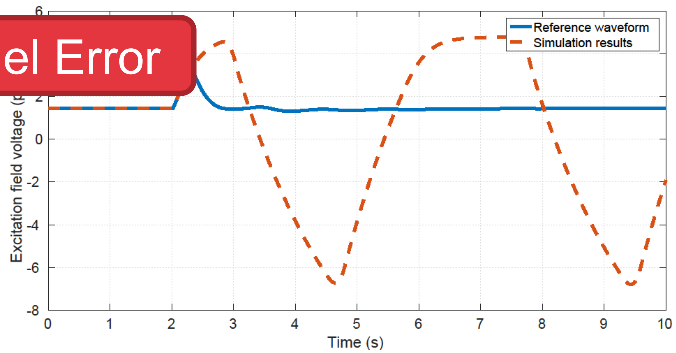
GitHub Integration

Syntax Error

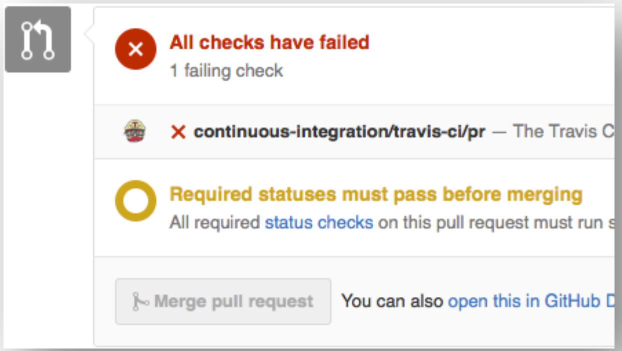
```
...penIPSL/Examples/Controls/PSSE/ES/IEEEX1.mo
...1:80:readonly] Error: Variable iEEEX1_1: In
   modifier (KA = 75), class or component KA not found
   in <OpenIPSL.Electrical.Controls.PSSE.ES.
   IEEEX1$iEEEX1_1>.
Error: Error occurred while flattening model OpenIPSL.
   Examples.Controls.PSSE.ES.IEEEX1
```

OR

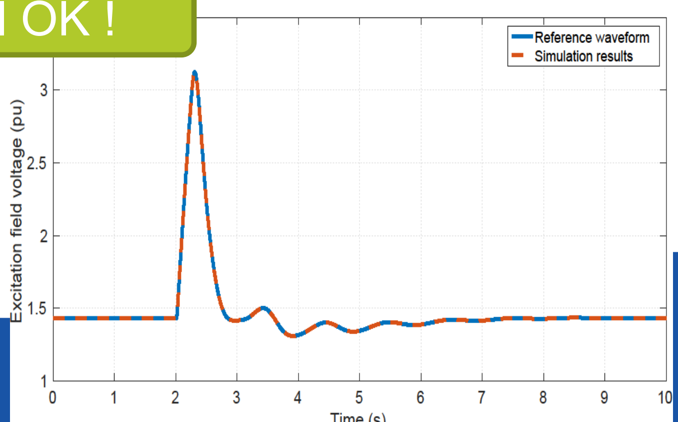
Model Error



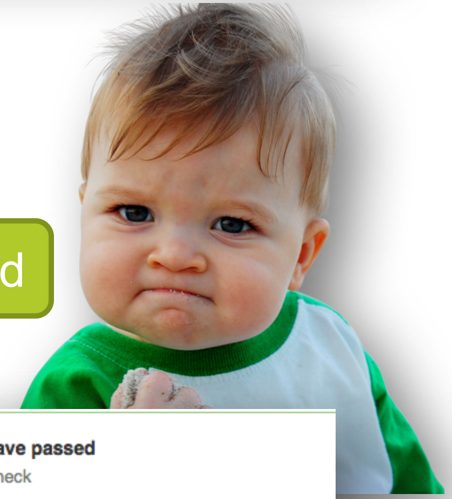
Merging Blocked

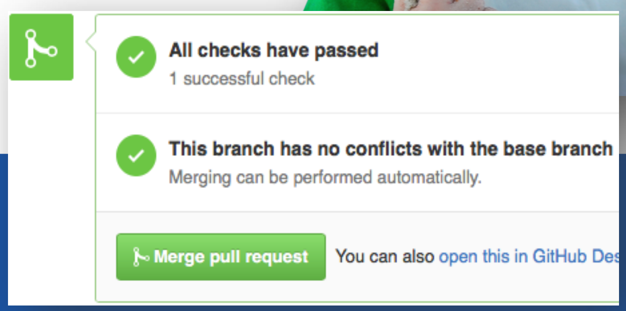


All OK !



Merging Allowed





Questions?

Main Take Away(s)

The implementation of Continuous Integration services allows to:

- Systematically check the code syntax
- Systematically check the integrity of the library (through SW-to-SW validation)

→ Easier collaboration with more developers

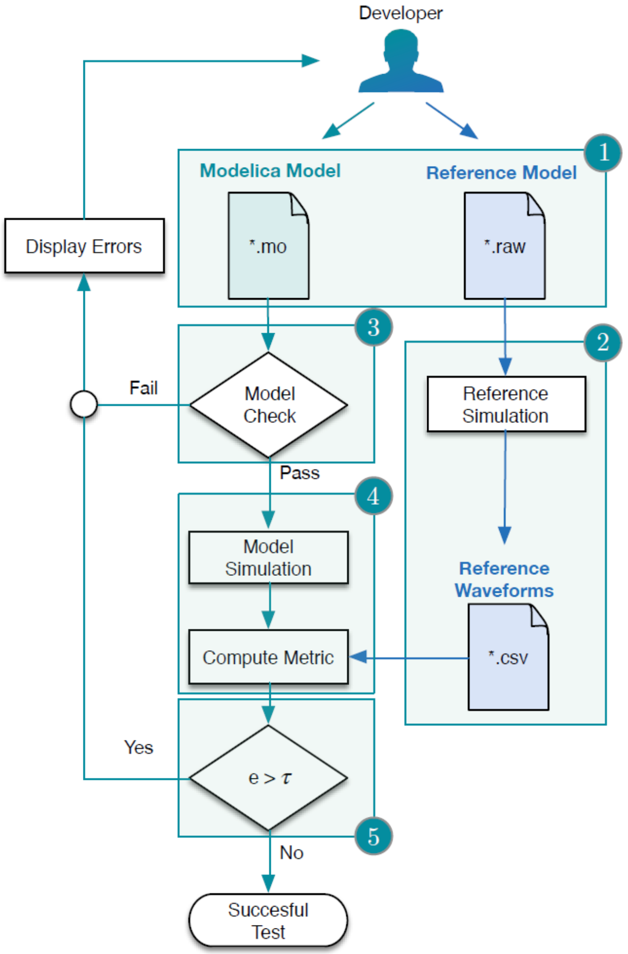
→ Easier to diagnostic potential errors

→ Better code quality

Other existing Modelica libraries could adopt CI:

→ Better compatibility with OM and

→ Modelica language version(s).



The **OpenIPSL** library can be found online: https://github.com/SmarTS-Lab/OpenIPSL

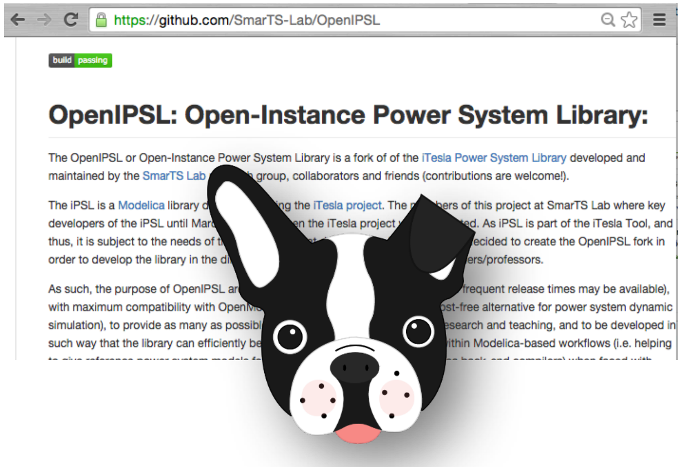
Let's now learn to use *OpenIPSL*!

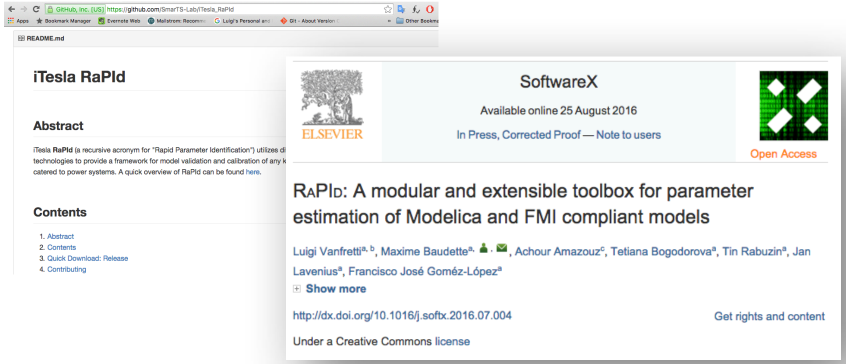
The **OpenIPSL** can be found online

- https://github.com/SmarTS-Lab/OpenIPSL

Our work on **OpenIPSL** has been published in the SoftwareX Journal:

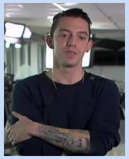
- http://dx.doi.org/10.1016/j.softx.2016.05.001





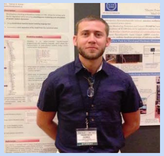
RaPId, a **system identification** software that uses OpenIPSL can be found at:

- https://github.com/SmarTS-Lab/iTesla_RaPId
- http://dx.doi.org/10.1016/j.softx.2016.07.004


Luigi Vanfretti

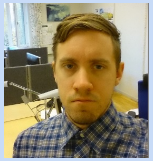

Achour Amazouz


Mohammed Ahsan Adib Murad


Francisco José Gómez


Giuseppe Laera


Tin Rabuzin


Jan Lavenius


Le Qi


Maxime Baudette


Mengjia Zhang


Tetiana Bogodorova


Joan Russiñol Mussons

Thanks to all current and former students and developers at

SmarTS Lab
Smart Transmission Systems Laboratory