SoftwareX 7 (2018) 34–36

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Software update

OpenIPSL: Open-Instance Power System Library — Update 1.5 to “iTesla Power Systems Library (iPSL): A Modelica library for phasor time-domain simulations"

Maxime Baudette [a], Marcelo Castro [b], Tin Rabuzin [a], Jan Lavenius [a], Tetiana Bogodorova [d], Luigi Vanfretti [c,*]

[a] *KTH Royal Institute of Technology, Stockholm, Sweden*
[b] *Universidade Federal de Juiz de Fora (UFJF), Juiz de Fora, Brazil*
[c] *Rensselaer Polytechnic Institute, Troy, NY, United States*
[d] *Ukrainian Catholic University, Faculty of Applied Sciences, Lviv, Ukraine*

ARTICLE INFO

Article history:
Received 4 December 2017
Received in revised form 11 January 2018
Accepted 11 January 2018

Keywords:
Modelica
Power systems
Simulation
Power system dynamics
Power system modeling

ABSTRACT

This paper presents the latest improvements implemented in the Open-Instance Power System Library (OpenIPSL). The OpenIPSL is a fork from the original iTesla Power Systems Library (iPSL) by some of the original developers of the iPSL. This fork’s motivation comes from the will of the authors to further develop the library with additional features tailored to research and teaching purposes. The enhancements include improvements to existing models, the addition of a new package of three phase models, and the implementation of automated tests through continuous integration.

Code metadata

Code metadata description	Please fill in this column
Current code version	v1.5.0
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00098
Legal Code License	MPL 2.0
Code versioning system used	git
Software code languages, tools, and services used	Modelica
Compilation requirements, operating environments & dependencies	Modelica Standard Library (MSL), Modelica IDE and corresponding C compiler
If available Link to developer documentation/manual	http://openipsl.readthedocs.io/en/latest
Support email for questions	luigi.vanfretti@gmail.com

1. Introduction

The original iTesla Power Systems Library (iPSL) is a Modelica library developed during the iTesla project [1]. After the completion of the iTesla project, the authors decided to independently pursue the development of the library with additional models and functionalities. The project was, thus, forked into the Open-Instance Power System Library (OpenIPSL). The main goal with this new library is to provide a framework more suitable for research and teaching activities, with the aim of establishing a transparent development approach. Hence, the discussions related to future developments of the project, and all improvements are documented on-line using the built-in functionalities offered by the *GitHub* hosting platform (e.g. bug-tracker, documentation). In addition, introductory tutorials have been developed and are now documented and accessible as releases on the on-line repository. These are important small steps towards establishing a user and developer community which the iPSL failed to nurture.

DOI of original article: http://dx.doi.org/10.1016/j.softx.2016.05.001.
* Corresponding author.
E-mail addresses: baudette@kth.se (M. Baudette), luigi.vanfretti@gmail.com (L. Vanfretti).

https://doi.org/10.1016/j.softx.2018.01.002
2352-7110/

Among the first modifications that were carried out in the OpenIPSL was to remove the models that were not functioning properly, and for which documentation was lacking. The remaining models were updated to be fully compatible with OpenModelica [2] to provide a free-of-cost alternative (as well as a fully open source software solution) for power system dynamic simulation. Additional improvements incorporated in this release include factorizing the existing code, taking advantage of the object-oriented syntax of the Modelica language adding new models and building software tools to facilitate the future maintenance of the project. Test models to showcase the updated models have also been added. These models are small-scale network models that can be used to validate each of the component's syntax, compilation and simulation results (i.e. regression testing).

Moreover, the library was updated with two entirely new features that are presented in this paper. First, we present an interface for three-phase modeling in Section 2 that has been implemented following the work of Taranto et al. in [3]. This interface allows one to combine positive-sequence transmission network models built using the OpenIPSL with three-phase distribution network models. Second, we present an implementation for automated testing in Section 3. This implementation is based on continuous integration software technologies [4] that facilitate and accelerate the development of quality code.

2. MonoTri: interfacing three-phase models

The library was built to model power system networks using the phasor (i.e. positive-sequence) modeling approach. This is the most common paradigm used to model large scale high voltage transmission networks, where assumptions of balanced conditions are valid. Hence, the use of the positive sequence phasors simplifies modeling of a power grid while preserving its most important characteristics. This approach, however, is not well-suited to model power distribution networks that are more prone to unbalanced conditions. Three-phase models are required in this case.

The computational burden of full three-phase models, including both the modeling of the "transmission" and the "distribution" parts of the network, is not negligible [5]. The solution proposed in [3], consists in a hybrid model including a single-phase representation for the "transmission" part of the network, and a three-phase representation for the "distribution" part. This is achieved through the implementation of an interface component, the *MonoTri* transformer, that interfaces three-phase models to the rest of the components from the library. Note that unlike co-simulation approaches or other interfaces [5], this interface is physically meaningful, modeling voltage and current flows explicitly. Therefore, the *MonoTri* models are actually a crucial part of the model of the systems' physics, and not a typical simulation signal transformation or exchange.

The implementation is currently in a separate Modelica package within the library. It contains the *MonoTri* transformer, a set of three-phase component models and some example models. The implementation was successfully validated against the original implementation in Simulight [6], using the IEEE four-node test feeder [7] and the IEEE 14-bus model, and the results have been submitted for publication [8].

3. Continuous integration implementation

The Authors participated in the development of the original iPSL project and experienced the limits of collaborating with no special organization or differentiated responsibilities. This motivated the team to adopt new software development practices based on well established software development methods. The main goal was to improve the code quality in the library, with a focus on fixing existing non-compiling models, while facilitating collaboration involving a larger development team to foster a durable community around the project.

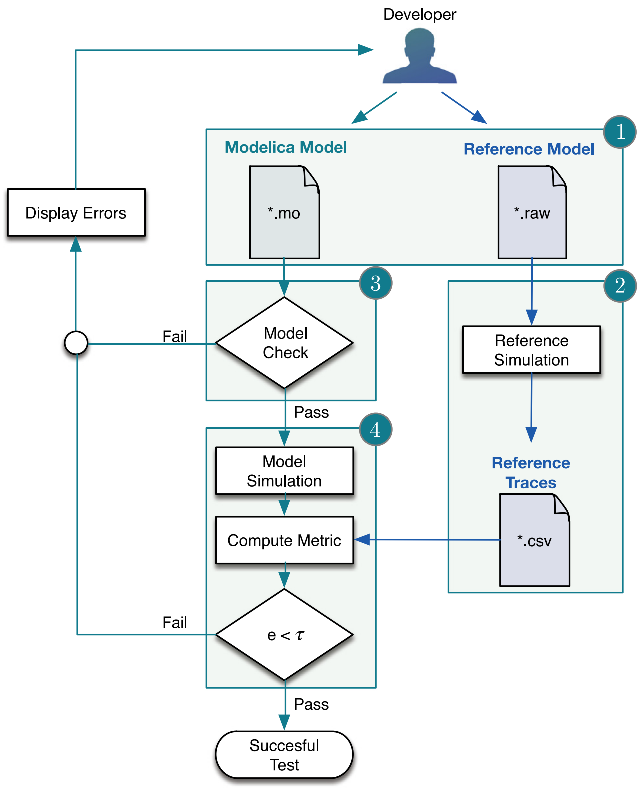

Fig. 1. CI Methodology.

The common approach to improve code quality in software development is to enforce systematic code testing and code review. The task can be implemented manually by adopting a code of conduct and strictly abiding to it. This solution can, however, easily be bypassed as it only relies on the good faith of the contributors.

The continuous integration approach seeks to address this issue by focusing on frequent, smaller, and targeted contributions that are continuously merged into the "production" version. Continuous integration services provide a set of tools to automate code testing, thus facilitating the code review task by taking care of repetitive tasks automatically.

The solution developed was implemented using Travis CI [4] configured to carry out a model check on the entire library to verify the compliance of the models to the Modelica syntax. The testing environment was built using a custom Docker image with OpenModelica and Python installed under a GNU/Linux distribution. The repository is configured to trigger Travis CI for each new pull-requested submission; the tests are run, and in the case where one model fails the test, an error is reported to the repository while at the same time blocking the merge of the pull-request. Finally, the repository is also configured to require a review of the submitted pull-request by at least one member of the maintenance team.

3.1. Extension for model regression testing

The solution, as presented above, was deployed in the repository and brought significantly better code quality in the existing code base. It is, for example, the main reason that brought the OpenModelica compatibility. In addition, the tests can be improved with a second stage using time-domain simulations to achieve behavior verification as well (i.e. regression testing).

The development of new models in OpenIPSL was validated through software-to-software validation by comparing time domain simulation against reference tools (e.g. PSS/E, PSAT) [9]. This task was manually executed for every newly developed models prior to including the models in the library. The goal was to automate the software-to-software validation process and to execute it after the code check stage. The resulting testing and validation methodology is depicted on Fig. 1. A prototype implementation was investigated and the results reported in [10].

4. Conclusion

The inception of the OpenIPSL project as an independent initiative has allowed the team of developers to implement the new features presented in this paper. Also, an important effort has been put into improving the code quality of the existing models. The resulting library is, thus, much easier to maintain.

The work towards this new version also consisted in improving the access to the library for the community. The documentation efforts through the development of tutorials seek to foster a durable user and developer community with the expectation to perpetuate the library's development. In that line, the upcoming development efforts will focus on implementing more automated tests to continue the simplification efforts in the collaborative process.

Acknowledgments

The voluntary contributions of some of the Authors and Dietmar Winkler (USN, Norway) is sincerely acknowledged. The support of the following funding bodies is gratefully acknowledged: Statnett SF, the Norwegian Transmission System Operator, and the EU funded FP7 project iTesla. The work of Luigi Vanfretti was primarily supported by the ERC Program of the National Science Foundation and DOE under NSF Award Number EEC-1041877. Other US government and industrial sponsors of CURENT research are also gratefully acknowledged.

References

[1] iTesla: Innovative tools for electrical system security within large areas. http:dx.doi.org/10.1109/pesgm.2014.6939447. URL http://www.itesla-project.eu/.
[2] Fritzson P, Aronsson P, Lundvall H, Nystrom K, Pop A, Saldmli L, Broman D. The openmodelica modeling, simulation, and software development environment. Simul Notes Eur 2005;(45):8–16.
[3] Marinho J, Taranto G. A hybrid three-phase single-phase power flow formulation. IEEE Trans Power Syst 2008;23(3):1063–70. http://dx.doi.org/10.1109/tpwrs.2008.922567.
[4] Hilton M, Tunnell T, Huang K, Marinov D, Dig D. Usage, costs, and benefits of continuous integration in open-source projects. In: Proc. 31st IEEE/ACM int. conf. automated software engineering; 2016. p. 426–37.
[5] Jalili-Marandi V, Ayres FJ, Ghahremani E, Belanger J, Lapointe V. A real-time dynamic simulation tool for transmission and distribution power systems. In: 2013 IEEE power & energy society general meeting. IEEE; 2013. http://dx.doi.org/10.1109/pesmg.2013.6672734.
[6] Assis TML, Taranto GN, Falcao DM, Manzoni A. Long and short-term dynamic simulations in distribution networks with the presence of distributed generation. In: 2006 IEEE power engineering society general meeting. IEEE; 2006. http://dx.doi.org/10.1109/PES.2006.1709545.
[7] Kersting WH. Radial distribution test feeders. In: Proc. (Cat. No.01CH37194) 2001 IEEE power engineering society winter meeting, vol. 2; 2001. p. 908–12. http://dx.doi.org/10.1109/PESW.2001.916993.
[8] Fernandes M de C, de Oliveira JG, Vanfretti L, Baudette M, Tomim MA. Modeling and simulation of a hybrid single-phase/three-phase system in modelica. In: VII simpósio brasileiro de sistemas elétricos, Niterói, RJ, Brazil; 2018 [submitted for publication].
[9] Zhang M, Baudette M, Lavenius J, Løvlund S, Vanfretti L. Modelica implementation and software-to-software validation of power system component models commonly used by nordic tsos for dynamic simulations. In: 56th conf. simul. model. Linkoping University Electronic Press; 2015. p. 105–12. http://dx.doi.org/10.3384/ecp15119105.
[10] Rabuzin T, Baudette M, Vanfretti L. Implementation of a continuous integration workflow for a power system modelica library. In: 2017 IEEE PES gen. meet; 2017.