



Introduction to

Open Source Software

(the promise for sustainable long-term synchrophasor software development)

Prof.Dr.-Ing. Luigi Vanfretti

E-mail: luigiv@kth.se, luigi.vanfretti@statnett.no

Web: <http://www.vanfretti.com>



SmarTS Lab
Smart Transmission Systems Laboratory

Statnett

luigiv@kth.se

Associate Professor, Docent
Electric Power Systems Dept.
KTH
Stockholm, Sweden

Luigi.Vanfretti@statnett.no

Special Advisor in Strategy and Public Affairs
Research and Development Division
Statnett SF
Oslo, Norway

- The information, views and opinions in this presentation are solely those of the author
- iTesla Project, Statnett SF, or KTH Royal Institute of Technology may not share the views of the contents of this presentation.
- Please keep in mind that I am a Univ. Professor...
 - I am just an *evangelist* for free and open source software
 - I am not a software engineer.
 - I am not a lawyer with IPR background.
 - I am not a business man.
 - ... there are certainly many people more knowledgeable than me about this topic.
- I am here to:
 - Share the knowledge I have gained over 10 years of using and contributing in OSS projects in academia.
 - Share my experience managing a small OSS project
 - Share my hope for sustainable long-term synchrophasor software development: so we can build on-top of the shoulder's of giants!

- Origins of OSS
- What is Open Source Software, Free Software and Proprietary Software?
 - Free (as in speech) / Libre software
 - Open Source Software
 - Proprietary (or closed source) Software
 - Key differences
 - Licensing
- Development of OSS – ‘The Cathedral and the Bazar’
- OSS and Business – benefits w.r.t. to Proprietary Software
- Myths about OSS
- OSS and Community
- Pathways to OSS Communities
- Efforts in the IEEE Task Force on OSS for Power Systems
- Planned OSS projects from the iTesla Project
- Conclusions and looking forward
- Addendum: Parallels with the history of software development for power system analysis



Origins of

Free/Libre and Open Source Software

- 1960s all software's source code was openly available
- 1970s (mid/late) proprietary software rises
 - A means to differentiate, and protect investments
 - A new and significant source of revenue
- 1984 Free Software Foundation established by Richard Stallman:
 - Out of frustration that the limitations of proprietary software imposed and driven by philosophical motivations: SW should be free as in freedom.
- 1998 Open Source Initiative (OSI) adopts OSS
 - To counteract negative connotations of **free software**
 - To reduce ambiguity of terms, as in: 'Free Speech, Free Beer'
- 2003 MITRE Study for US Department of Defense (DoD)
 - Found OSS more widely used in critical roles than expected
- 2009 DoD [Memorandum](#) "Clarifying Guidance..."
 - In almost all cases, OSS meets the definition of "commercial computer software" and shall be given appropriate statutory preference in accordance with 10 USC 2377

- One can divide software in three categories depending on how they are distributed (licensing terms) and developed:
 - Proprietary Software
 - Free/*Libre* Software (FS)
 - Open Source Software (OSS)
 - Free/*Libre* and Open Source Software (FLOSS) is a term used to merge common characteristics of Free/*Libre* Software & Open Source Software



Proprietary Software

Definition

- Proprietary SW has restrictions for:
 - Its use, modifications, and, more notably, on copying, distribution, or publishing of unmodified or modified versions (if even possible)
 - The restrictions are placed by the owners of the software and detailed in a *license agreement* (EULA)
 - In the USA, copyright laws provide severe penalties for unlawful distribution of copyrighted material
 - Reverse engineering of the SW could also violate the US Copyright Law or the Digital Millennium Copyright Act
 - The term *closed software* has been suggested to refer to all this type of SW and avoid the derogatory misunderstanding of other terms in use.



Open Source Software (OSS)

Definition

- OSS is one that complies with the Open Source Definition (OSD), published by the Open Source Initiative (OSI).
- The OSD gives the criteria to which SW wishing to adopt an OSS license must comply with.
- The OSI maintains a list of OSS licenses.

Open Source Definition (derived from the Debian FS Guidelines):

Right #1: Free (liberty) redistribution.

Right #2: Source code available.

Right #3: Derived works permitted.

Right #4: Integrity of the author's source code.

Right #5: No discrimination against persons or groups.

Right #6: No discrimination against fields of endeavor.

Right #7: Distribution of license.

Right #8: License must not be specific to a product.

Right #9: License must not contaminate other SW.



Free/Libre Software (FS)

Definition

- A term coined by Richard Stallman.
- FS is software that can be copied and distributed in modified or unmodified form without restrictions.
- Restrictions may be used only to ensure that future recipients of the software can also copy, study, modify and distribute the software – this is the concept of copyleft; and main difference with OSS.

Tenants of the FS movement:

Freedom 0: to run the program, for any purpose.

Freedom 1: to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition.

Freedom 2: to redistribute copies so you can help your neighbor.

Freedom 3: to distribute copies of your modified versions to others.

Key Difference

a delicate variance in philosophy

- Philosophical:
 - Open source shares the philosophical orientation of the proprietary approach, but rejects its techniques
 - Free software advocates reject the orientation and logic of both proprietary and open source approaches
- The Free Software movement is a campaign for computer users' freedom – a nonfree program is an injustice to the user.
- The Open Source movement does not have this ethical position, and focuses on the practical benefits and value that can be created from “seeing the source code”.
- Indeed, the OSS movement originated as a schism of the FS community:
 - OSS ignores and detaches itself from the Free SW movement's ethical and social positions
 - OSS focuses on the practical values - having powerful and reliable SW
- *In practical terms:*
 - *while the OSS principles see the possibility of co-existing and co-operating with free software and non-free software, the FSS principles reject this possibility (all software should be free as in freedom).*



Free/Libre and Open Source Software (FLOSS)

Definition

- From the perspective of an OSS or FS advocate, **focusing on practical values**, the two approaches seem identical.
 - FLOSS is a merger of FS and OSS concepts focusing on their common software development and distribution characteristics.
- This aggregation ignores the subtle, **but transcendental philosophical** differences between FS and OSS.
- The term is used to show neutrality:
 - Shows no preference between any of the political philosophies of the OSS or FS camps.
- If you don't want to have any association with the philosophy of FSF, I recommend you use the term OSS:
 - In the words of Eric Raymond: using the “free software” is lousy marketing, it's not something that businesses want to hear.



Software Licensing

- Licenses reflect the terms that the originators decided to impose on the users (and modifiers of the work).
- **Free Licenses:**
 - or reciprocal, it's provisions require that on re-licensing the source code must remain open.
 - If you choose to distribute an application you programmed or modified, you must include the source code
 - Most important: GNU Public License (GPL).
 - Also called “viral”.
- **Open Licences:**
 - Do not contain provisions on re-licensing.
 - They allow derivative works from OSS to be converted into closed SW
 - They are “non-reciprocal”: the primary recipient obtains the source code but it may not pass it on.
 - OSI maintains a list of Open Licenses (BSD and MIT are examples)
- **Non-Open Source Licenses:**
 - Classic proprietary licenses.
 - There is no distinction between source code and binary, because the source code is not distributed.
 - Re-distribution is prohibited.
- **Other licenses:** Qt License, Artistic License, Creative Commons ... etc., have very unique features.



The Cathedral and the Bazaar

SW Origins Development Methods

The Bazaar Model Characteristics:

Users are treated as co-developers: “Given enough eyeballs all bugs are shallow”

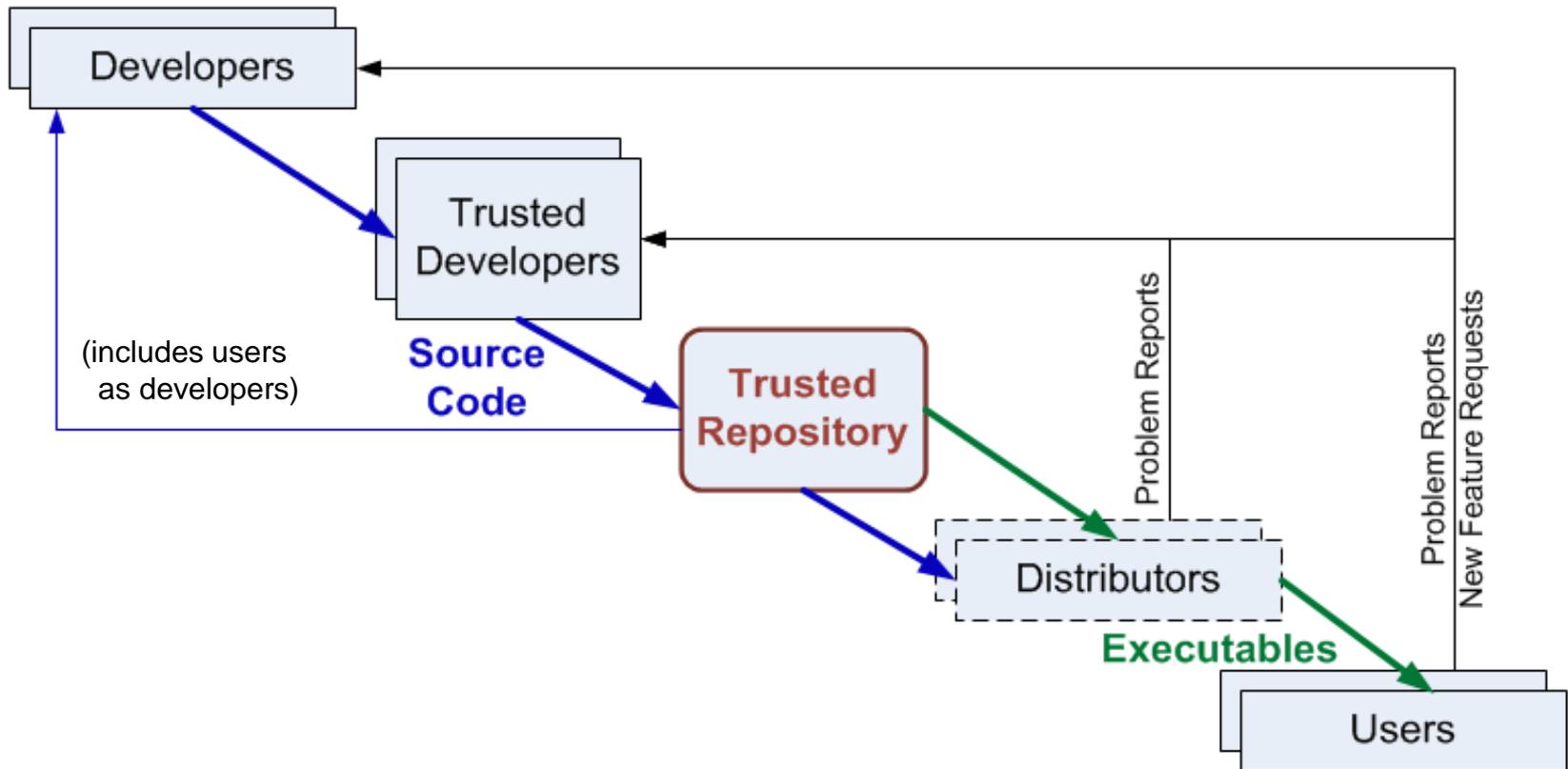
Early releases: increases the chances of finding co-developers.

Frequent integration: code changes should be merged into the code base as soon as possible to avoid overheads in fixing a large number of bugs between releases. OSS projects use nightly builds when integration is done automatically.

Several Versions: stable and buggy version. Stable has fewer features, but doesn't break. Development version has bugs, and user's take the risk of using the code, which in turn helps to identify and fix bugs.

High modularization: the SW should be modular to allow parallel development on different components.

Dynamic decision making structure: formal or informal, takes decisions depending on user requirements or technological issues.



- From David A. Wheeler Presentation, 11/4/2009
- **More from Russell about this.**



OSS and Business

benefits w.r.t. proprietary software

- Proprietary SW
 - Support is a monopoly: only one company has the source code
 - Only one company can provide support, and the support may not be of good quality
- The first OSS business was Cygnus Software originating from Stanford's Electronics Research Lab
 - The ideas on how to make money out of OSS came from the GNU manifesto
 - The idea was to sell consulting and services around the GNU Free Software.
 - They developed a model that could provide 2-4 times the support and handholding capability for $\frac{1}{2}$ to $\frac{1}{4}$ the cost.
- ***OSS allows for a free market*** for any kind of service or support:
 - If you use OSS in your business and you want good support, you have a **choice** of entities that can provide good support
 - These business will have to provide good support or you will go to someone else.

Myths about OSS

- OSS is less secure: “A security system is only as secure as its secret. Beware pseudo-secrets.”
 - Claim: OSS is insecure because people can get their hands on the code.
 - The proprietary approach of “security through obscurity” is as ineffective for protecting SW – availability of source code has little or no bearing on the security of the software.
- OSS has low quality and low performance
 - OSS has a larger community interested in the development of the SW itself, not just one vendor.
 - Peer review and the fact that these communities are invested in quality (and not making a buck from every functional improvement)
- Most OSS developers are college students
 - Maybe 20 years ago... there is large number of open source software vendors, and in many areas it is increasing due to the demand from prospective users to avoid the restrictive nature of the proprietary model.
 - This proposition is an out of date myth based on the lack of understanding about modern day priorities and requirements of users.
- OSS is unsupported
 - Did you know Google and Facebook rely on Linux? These players are concern in maintaining SW that runs the platform on which businesses are built!
 - With OSS you always have the **choice** of bringing an independent third party for consultation... and not having to wait for the proprietary vendor to take care of your problem.
- OSS has zero cost
 - Nope! An OSS vendor can charge you for the SW license and still be open source. There is no relation between the type of license and what you have to pay to get it.

OSS and Community

- Community is vital to an OSS project – an OSS license is not enough to bring users and developers to a project and build a community.
- OSS originates because someone wants something built or to respond to meet future needs of others (product development).
- A community originates from a group of individuals sharing common interests - most of which will be passive, but others choose to have more active roles:
 - Reporting bugs, helping other users, writing documentation or evangelizing.
 - Active members can be rewarded for their efforts (e.g. additional access or control over a project, or simply “peer recognition and appraisal”)



Paths for an OSS Community

- “Release early. Release often.” and listen to your users: attracts persons that can contribute.
- Developers are needed: and they can be attracted from the user base or from elsewhere (people interested by the technical challenge)
- Presentation and branding: to convince prospective users that the SW does better than others.
- Get organized:
 - Make the SW understandable, set up dev web site, email lists – **and write documentation!!!**
- Avoid benevolent dictatorships:
 - A benevolent dictator is a single person in charge of major new functionalities and reviewing contributed code.
 - This alienates developers who need to be communicated the importance that they “can do more in concert than individually”.
 - Developers will only remain if the leader can make the project a place that they want to keep coming back to – a reward system is always needed.
- OSS communities must have the ability to outlive their founder’s original interest:
 - If they rely heavily on a benevolent dictator, they will fragment and fall apart when the leaders move or retire
 - It is important to ensure that when key developers move on, their roles are adopted by others.
- A governance model:
 - That captures the shared understanding and goals of the OSS – is needed in documented form.
 - This ensures the community has a life of its own that can survive as long as there is a sustained need for the project’s outputs.

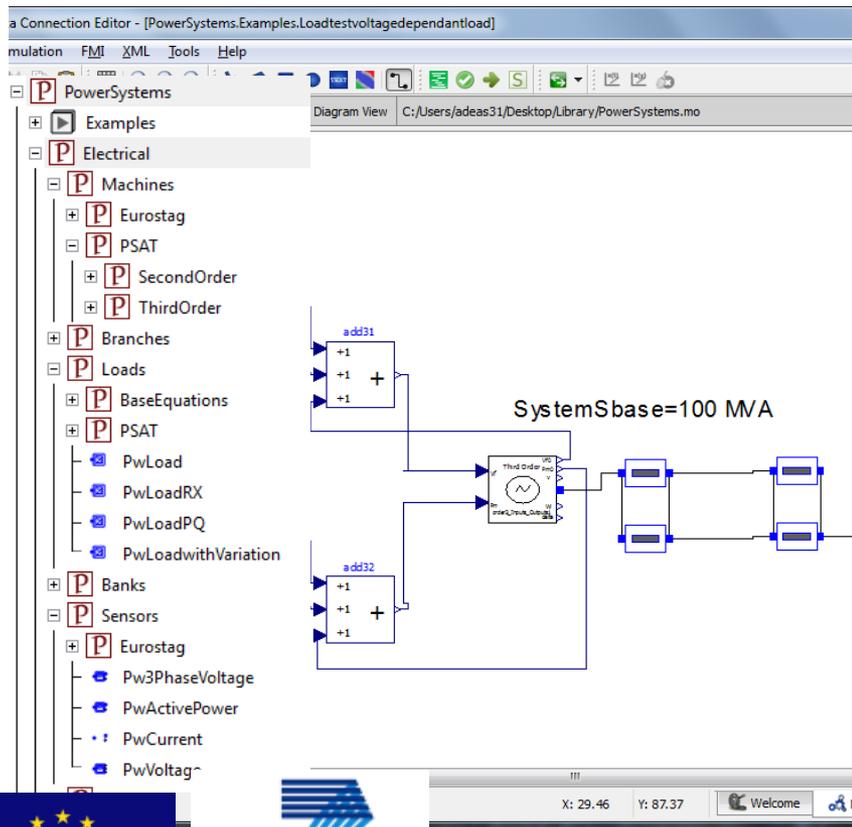


IEEE PES TF on OSS for Power Systems

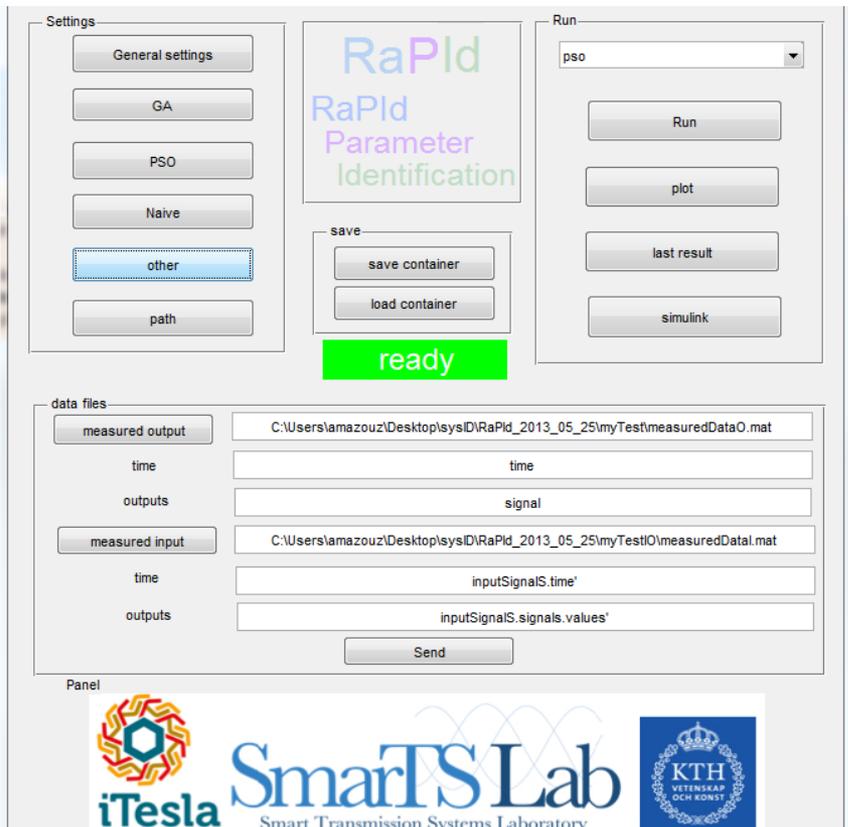
- http://ewh.ieee.org/cmte/pspace/CAMS_taskforce/format.htm
- Explores the potential for FLOSS in the Power & Energy Society
- Panel sessions on FLOSS have been organized in Montreal 2006, Tampa 2007, Calgary 2009, Detroit 2011 and Vancouver 2013.
- The TF formally initiated activities in Fall '08, with 28 members currently registered.
- Mission:
 - (i) to diffuse the philosophy of FOSS in the power systems community; and
 - (ii) promote FOSS for the benefit of the PES ranging from pedagogical purposes to commercial-grade applications.
- The web page collects a list of free and/or open source software packages for power system analysis.
- To better diffuse the FLOSS philosophy, the TF is preparing a report on the state of the art of FLOSS for power systems analysis (mostly on simulation software)
- New members, ideas and **leadership** are desirable...

Planned Open Source Software to be Released

iTesla Power Systems Modelica Library



iTesla Rapid Parameter Identification Toolbox






Conclusions

- *Education and research takes advantage from OSS:*
 - In my experience, the students generally accept with enthusiasm the ideas from OSS and tend to diffuse them once they begin their professional career.
 - However, to achieve this goal, professors should adopt OSS ideas first.
- *OSS is a business:*
 - OSS should not be limited to universities or idealists, it can also be a business.
 - Several open source projects have become important companies.
 - I believe that there is room for this kind of business also in power systems.
 - Projects as OpenDSS is used as a platform for other EPRI's consulting services.
 - OpenPDC is another great example of how OSS has a clear potential to enable innovation and build a software service market in power systems.
- *OSS reduces costs and improves reliability:*
 - OSS offers advantages that could incentive companies to provide their products under a GPL license (or other) is the fact that OSS projects typically have a large number of users.
 - This basically provides an invaluable testing platform.
 - One of the most important issues of software applications for power system analysis is that the users have no actual way of contributing towards the development or even to be able to fix their own issues when they are capable to do so; this does not make SW reliable.

Looking Forward

- The future of OSS for synchrophasors is promising although not completely clear.
 - One of the most difficult tasks is to motivate researchers and developers to share their knowledge and code
 - Something that is less common in the power community as compared to other communities.
 - **We need to create the context to *have an actual community* around the SW**
- We need to transmit practitioners the idea that OSS is reliable and can be a business, as several OSS projects have largely demonstrated.
 - The product is the know-how and the experience of the people that maintain the software, not the software tool itself.
 - The flexibility and versatility of OSS projects can be a key feature of the success of emerging smart grid technologies.
- **The future of FOSS for synchrophasors depends on you!:**

Share, and be rewarded tenfold!

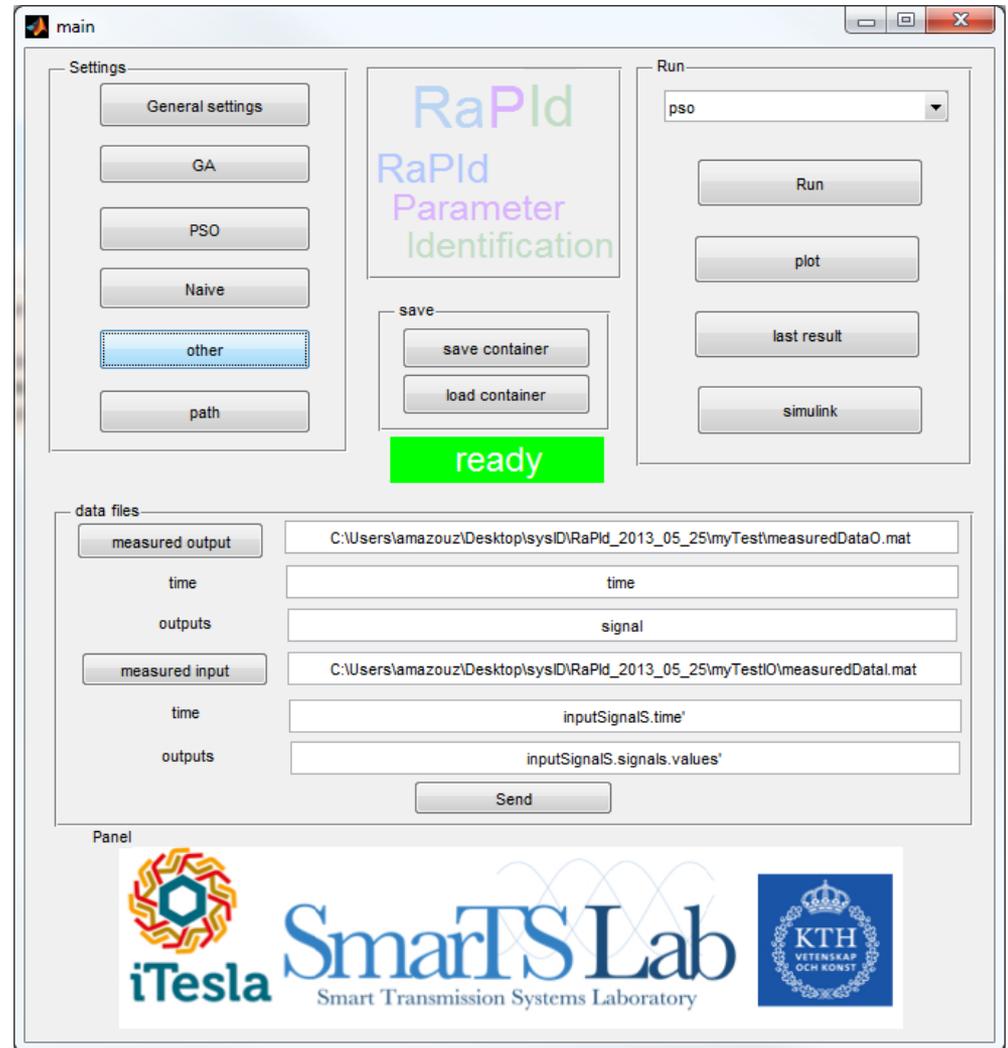
- For the philosophers and futurists:
 - Richard M. Stallman, *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Boston: Free Software Foundation, 2002.
 - S. Chopra and S. D. Dexter, *Decoding Liberation: The Promise of Free and Open Source Software*. New York: Routledge Taylor & Francis Group, 2008.
- For the developers and anthropologists:
 - Eric S. Raymond, *The Cathedral and the Bazaar*. Thyrus Enterprises, 2000. Available at: <http://www.tuxedo.org/>
- For the lawyers:
 - R. M. Stallman, *GNU General Public License*. Free Software Foundation, 2007, available at <http://www.gnu.org/copyleft/gpl.html>.
 - K. Coar, "Open Source Definition," 2007, available at <http://www.opensource.org/docs/osd>
- For the generalists:
 - J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani, Eds., *Perspectives on Free and Open Source Software*. Cambridge, MA: MIT Press, 2005.
 - C. DiBona, S. Ockman, and M. Stone, Eds., *Open Sources: Voices from the Open Source Revolution*. Sebastopol, CA: O'Reilly & Associates, 1999.
 - C. DiBona, D. Cooper, and M. Stone, Eds., *Open Sources 2.0: The Continuing Evolution*. Sebastopol, CA: O'Reilly Media, 2006.
- For the licensors:
 - A. M. S. Laurent, *Understanding Open Source & Free Software Licensing*. Sebastopol, CA: O'Reilly Media, 2004.
 - A.G. Gonzalez, "The software patent debate," *Journal of Intellectual Property Law & Practice*, vol. 1, no. 3, pp. 196–2006, Jan 2006.
 - R. W. Hahn, Ed., *Government Policy toward Open Source Software*. Washington, D.C.: AEI-Brookings Joint Center for Regulatory Studies, 2002.
- For the business persons:
 - P. Kavanagh, *Open Source Software: Implementation and Management*. Oxford, UK: Elsevier Digital Press, 2004.
 - D. Woods and G. Guliani, *Open Source for the Enterprise: Managing Risks, Reaping Rewards*. Sebastopol, CA: O'Reilly Media, 2005.
- For the power system analysts:
 - S. Fustar, "The Impact of Open Source Software on the Next Generation of Energy Systems," *IEEE PES Gen. Meeting*, June 2007.
 - F. Milano and L. Vanfretti, "State of the Art and Future of OSS for Power Systems," in *IEEE PES Gen. Meeting*, Calgary, Canada, Jul. 2009.

Thank you!

Questions?

luigiv@kth.se

luigi.vanfretti@statnett.no



The screenshot shows the RaPIId software interface, titled "main". The interface is divided into several sections:

- Settings:** A vertical stack of buttons for "General settings", "GA", "PSO", "Naive", "other" (highlighted with a blue border), and "path".
- Central Panel:** Displays the text "RaPIId Parameter Identification" in a stylized font. Below it, there is a "save" section with "save container" and "load container" buttons, and a prominent green "ready" button.
- Run:** A dropdown menu set to "pso", followed by buttons for "Run", "plot", "last result", and "simulink".
- data files:** A section for file paths. It includes "measured output" (C:\Users\lamazouz\Desktop\sysID\RaPIId_2013_05_25\myTest\measuredData0.mat), "time" (empty), "outputs" (empty), "measured input" (C:\Users\lamazouz\Desktop\sysID\RaPIId_2013_05_25\myTest\measuredData1.mat), "time" (inputSignalS.time'), and "outputs" (inputSignalS.signals.values'). A "Send" button is located at the bottom of this section.
- Panel:** A footer area containing the iTesla logo, the "SmarTS Lab" logo (Smart Transmission Systems Laboratory), and the KTH logo (KTH VETENSKAP OCH KONST).



The Parallel History of *power system analysis software*

- Back in the 60s & 70s, all scientific communities were in the same condition: most software was open source *de facto* and was shared among experts in the area.
 - Software for power flow and transient stability became available around mid 60s.
 - Programs ran in mainframes, GE and Westinghouse were the main service providers.
 - Large companies that had mainframes (for billing) started looking into using them for power system studies.
 - By the late 60s many utilities in the USA had developed their own power flow and stability programs: Philadelphia Electric Co. (PECO) and BPA's became widely used programs for planning.
 - These programs **and their source code** were freely given away (the term "open source" did not exist yet), and the BPA SW was in the public domain because it was developed by a US gov. entity.
 - BPA and PECO had well-known groups of power engineers who developed, maintained and improved the SW throughout the 70s and into the 80s.
 - Other power companies that used these software, did not had their own groups to support it and BPA and PECO could not provide support.
 - Thus, vendors of planning SW who could provide such user support also thrived in parallel.
 - By the late 80s even PECO and BPA decided to disband their in-house expertise in SW development and the use of these packages dwindled.
 - There are few traces of these programs left, except for their mention in the technical literature from those days.



Lessons from the Parallel History of *power system analysis software*

- In the early stages of power system software development, there was no concept of “open source software” and so, there was no understanding of the importance and consequences of sharing with the community the code implemented.
- After the paradigm of *proprietary software* appeared, companies locked out code and imposed their products by different lock-in methods (e.g. model data format)
- The main reason that relatively few OSS SW are available today for power system analysis (mainly academic research projects or government funded efforts) is that:

Let's learn from history – and not make
the same mistakes again!

- For OSS to succeed in the synchrophasor business we need to **create a service market, develop & implement OSS that supports standards and develop expertise of engineers to transition them from users to SW authors (early on!)**